Algorithms and Hardness Results for Compressing Graphs with Distance Constraints

by

Zeyu Zhang

A dissertation submitted to The Johns Hopkins University

in conformity with the requirements for the degree of

Doctor of Philosophy

Baltimore, Maryland

March, 2019

Abstract

Graphs have been widely utilized in network design and other applications. A natural question is, can we keep as few edges of the original graph as possible, but still make sure that the vertices are connected within certain distance constraints.

In this thesis, we will consider different versions of graph compression problems, including graph spanners, approximate distance oracles, and Steiner networks. Since these problems are all NP-hard problems, we will mostly focus on designing approximation algorithms and proving inapproximability results.

Thesis Committee

Primary Readers

Michael Dinitz (Primary Advisor)
Assistant Professor
Department of Computer Science
Johns Hopkins Whiting School of Engineering

Xin Li
Assistant Professor
Department of Computer Science
Johns Hopkins Whiting School of Engineering

Amitabh Basu
Associate Professor
Department of Applied Mathematics and Statistics
Johns Hopkins Whiting School of Engineering

Vladimir Braverman
Associate Professor
Department of Computer Science
Johns Hopkins Whiting School of Engineering

Alternate Readers

Abhishek Jain
Assistant Professor
Department of Computer Science
Johns Hopkins Whiting School of Engineering

Acknowledgments

I would like to express my sincere gratitude to my advisor Michael Dinitz for the continuous support of my Ph.D. study and research. This thesis is mostly a collection of our joint works. He is always full of ideas, and has good intuitions on the problems we are working on.

During my study I also worked with Prof. Vladimir Braverman and Prof. Xin Li, as well as Amy Babay, Guo Chen, Kuan Cheng, Aditya Krishnan, Yasamin Nazari, and Lin Yang. Although most of the work has not been published yet, I still learned a lot and am really appreciative for their help.

I would also like to thank my parents, who brought me up and supported my entire life.

Table of Contents

List of Algorithms

List of Figures

Chapter 1

Introduction

Graph compression has been intensively studied and widely used in different areas. Compressing a graph not only allows us to reduce the storage space in memory, but it also allows algorithms to run faster because the size of the problem decreased.

There are multiple ways to compress a graph. In some circumstances, we are not allowed to lose any information. However, information theory tells us that in this case we must still use $\Omega(n^2)$ bits in average. Therefore, in order to have much smaller graphs, information loss is unavoidable. Since the distance between pairs of vertices in the graph is often the major thing we care about, we will mainly focus on the case that some increase of the distance between pairs of vertices is allowed, as long as it is bounded.

Spanners are one of the most interesting types of graph compression. A k-spanner (or a spanner with stretch k) of a graph G is a subgraph, where distances in this subgraph are at most k times the original graph. They were originally introduced by Peleg and Schäffer (Peleg and Schäffer, 1989) and Peleg and Ullman (Peleg and Ullman, 1989) in the context of distributed

computing, but can also be used in compact routing (e.g. Thorup and Zwick, 2001), property testing (e.g. Bhattacharyya et al., 2009), and preprocessing approximation algorithms for graphs. The fundamental result in this area is given by Althöfer et al. (Althöfer et al., 1993), which basically says that every graph has a $(2t-1)$-spanner with at most $n^{1+\frac{1}{t}}$ edges. There are many different settings and hundreds of papers extending the result of this work (e.g., studying the tradeoff between the stretch and the total weight (Chandra et al., 1992) rather than the stretch and the size).

In parallel to these works on tradeoffs between the stretch and the size, there has been a line of work on optimizing spanners (Kortsarz and Peleg, 1994; Berman et al., 2013; Dinitz and Krauthgamer, 2011a). In this setting we are given a graph G and a stretch bound k, and the objective is to construct a k-spanner, with size that is comparable to the size of the best possible k-spanner of G. The first result of this thesis is on approximating low stretch spanners, which will be introduced in Section 1.1, and the technical details appear in Chapter 2.

Another interesting type of graph compression is approximate distance oracle. Given a finite metric space (V,d), an approximate distance oracle is a data structure which, when queried on two points $u,v \in V$, returns an approximation to the actual distance between u and v which is within some bounded stretch factor of the true distance. The seminal work for this problem is due to Thorup and Zwick (Thorup and Zwick, 2005). They proved that every metric space has a approximate distance oracle with stretch $(2t-1)$ using $O(tn^{1+\frac{1}{t}})$ space and query time $O(t)$.

We can see that this tradeoff between the stretch and the size of the data structure is quite similar to the tradeoff on spanners. It seems straightforward to also take an optimization view on approximate distance oracles, similar to what we and others have done for the spanner problem: given a metric space (V, d), can we find (or approximate) the smallest approximate distance oracle on (V, d)?

However, it is not clear whether this question is even well-defined. Lower bounds on data structures are commonly arrived at through information or communication complexity (see, e.g., Jacobson, 1988), but when we ask for the optimal data structure on one particular instance this approach becomes meaningless: for any specific instance, there is a data structure which uses size $O(1)$ on this instance, by storing it in the query algorithm. In Section 1.2, we will introduce the ideas on approximating approximate distance oracles, which shows how to get around of this issue by restricting attention to certain classes of distance oracles, as well as giving algorithmic results. The technical details will be in Chapter 3.

We can see that both definitions of spanners and approximate distance oracles require that the distance constraints to be some function of the original distance, but what if we loosen the constraint, and just ask them to be connected, or to maintain a global distance bound? It becomes more or less a STEINER NETWORK problem.

In the simplest case, if the demands are all pairs and the vertices just need to be connected, then this is the classical MINIMUM SPANNING TREE problem. If we consider other classes of demands, then we get more difficult but still

classical problems. Most notably, if the demands form a star (or any connected graph on V), then we have the famous STEINER TREE problem. If the demands are completely arbitrary, then we have the STEINER FOREST problem. Both problems are known to be in FPT parameterized by the number of demands (Dreyfus and Wagner, 1971) (i.e., they can be solved in $f(p) \cdot poly(n)$ time for some function f, where p is the number of demand pairs).

There are many obvious generalizations of STEINER TREE and STEINER FOREST. For example, DIRECTED STEINER TREE (DST) and DIRECTED STEINER NETWORK (DSN) are just considering these problems on directed graphs. Both problems have been well-studied (e.g., (Charikar et al., 1999; Zosin and Khuller, 2002; Chekuri et al., 2011; Chlamtác et al., 2017; Abboud and Bodwin, 2018)), and in particular it is known that the same basic dynamic programming algorithm used for STEINER TREE will also give an FPT algorithm for DST. However, DSN is known to be W[1]-hard, so it is not believed to be in FPT (Feldmann and Marx, 2016).

This gives an obvious set of questions: what demand graphs make the problem "easy" (in FPT) and what demand graphs make the problem "hard" (W[1]-hard)? Recently, Feldmann and Marx (Feldmann and Marx, 2016) completely characterize the "easy" and "hard" cases for DSN. This inspires us to think about the same question of the distance-bounded variations of STEINER TREE and STEINER FOREST.

In Section 1.3, we will introduce the SHALLOW-LIGHT STEINER TREE (SLST) problem and SHALLOW-LIGHT STEINER NETWORK (SLSN) problem. This is setting that the distance constraint is a global constant, where for

SHALLOW-LIGHT STEINER TREE the demand graph is a star, and SHALLOW-LIGHT STEINER NETWORK has arbitrary demands. We will also show how we characterize "easy" and "hard" cases for these problems. The technical details are in Chapter 4.

1.1 Spanners

A k-spanner of a graph $G = (V, E)$ is a subgraph H of G in which $d_H(u, v) \leq k \cdot d_G(u, v)$ for all $u, v \in V$, where d_H and d_G denote shortest path distances in H and G, respectively. The factor k is also called the stretch of the spanner.

The fundamental tradeoff between the number of edges and the stretch given by (Althöfer et al., 1993) shows that, for all integer $t \geq 1$, there is a $(2t-1)$ spanner with at most $n^{1+\frac{1}{t}}$ edges for any undirected graph G. This result is given by a simple greedy algorithm, which first sorts all the edges, starting from the shortest one, and picks an edge (u, v) into H if (u, v) does not already satisfy the distance constraint (i.e. $d_H(u, v) > (2t-1) \cdot d_G(u, v)$).

There is a long standing conjecture due to Erdős which says that there exist graphs with $\Omega(n^{1+\frac{1}{t}})$ edges and girth $2t+1$ for all t and large enough n. This is known as the Erdős girth conjecture. If we assume this conjecture holds, then the fundamental tradeoff is actually tight, because the only $(2t-1)$ spanner of these graphs are themselves. Here "girth" is the size of the smallest cycle in the graph.

It seems like there is nothing we can do because the upper and lower bound match each other. However, we can take an optimization view of this problem, where we are given a graph G with a stretch bound k, and the

objective is to construct the smallest possible k-spanner for G. This problem is called BASIC k-SPANNER in the undirected case, and DIRECTED k-SPANNER in the directed case. The fundamental tradeoff only shows an $O(n^{\frac{1}{\lfloor (k+1)/2 \rfloor}})$-approximation to BASIC k-SPANNER, because the best k-spanner can be as small as $\Theta(n)$ edges. Such a large polynomial approximation ratio seems quite weak, which motivates recent work on optimizing spanners: can we get better approximation ratio than the fundamental tradeoff?

It is particularly interesting to focus on small values of k. One of the reasons is that spanners are most useful when the stretch is small. Another reason is that when $k = \Omega(\log n)$, we have $O(1)$-approximation by just using the fundamental tradeoff bound, which means there is little space for improvements.

There is a classic greedy algorithm given by Kortsarz and Peleg (Kortsarz and Peleg, 1994) for BASIC 2-SPANNER, which gives an $O(\log n)$-approximation. Later Berman et al. (Berman et al., 2013) used linear programming and presented an $\tilde{O}(n^{\frac{1}{3}})$-approximation in the $k = 3$ case, where the fundamental tradeoff only gives an $O(\sqrt{n})$-approximation. Recently, our main result in (Dinitz and Zhang, 2016) extends the previous results to the $k = 4$ case and gives an $\tilde{O}(n^{\frac{1}{3}})$-approximation algorithm, where the previous best is also the fundamental tradeoff with $O(\sqrt{n})$-approximation.

For the lower bounds, it was shown recently that BASIC k-SPANNER is actually LABEL COVER-hard, which means, unless NP $\subseteq$ BPTIME$(2^{\text{polylog}(n)})$, there is no polynomial-time algorithm for BASIC k-SPANNER with $2^{(\log^{1-\varepsilon} n)/k}$ approximation factor for any constant $\varepsilon > 0$ and $k > 2$ (Dinitz, Kortsarz,

and Raz, 2012). However, up to now there is no polynomial hardness result known for this problem. Based on the observation that almost all non-trivial upper bounds for optimizing spanners are given by a standard flow-based LP, proving the existence of a polynomial integrality gap can help us understand the limitation of this LP approach. Our second result basically proved an $\Omega(n^{\frac{1}{\lfloor \frac{k+1}{2} \rfloor + 2}})$ integrality gap, which is close to the fundamental tradeoff.

There is also an interesting extension called fault-tolerant spanners, which was first introduced by (Levcopoulos, Narasimhan, and Smid, 1998) in the geometric graph case. In this setting we want the subgraph to be a spanner even when vertex failures happen. Chechik et al. (Chechik et al., 2010) then extended this concept to general graphs, and (Dinitz and Krauthgamer, 2011a; Dinitz and Krauthgamer, 2011b) studied the optimization version of this problem.

A subgraph H of G is an f-fault-tolerant k-spanner if $d_{H \setminus F}(u, v) \leq k \cdot d_{G \setminus F}(u, v)$ for all $u, v \in V$ and $F \subseteq V$ with $|F| \leq f$. In other words, $H \setminus F$ must be a k-spanner of $G \setminus F$ for all sets $F \subseteq V$ with size at most f. The optimization problem is that we are given a graph G and integers k, f, and the objective is to output an f-fault-tolerant k-spanner of G with as few edges as possible.

Previously, all the algorithms (Chechik et al., 2010; Dinitz and Krauthgamer, 2011a; Dinitz and Krauthgamer, 2011b) for approximating f-fault-tolerant k-spanner have high dependency of f in the approximation ratio, except for the $k = 2$ case where (Dinitz and Krauthgamer, 2011b) gave an $O(\log n)$-approximation. While in absolute terms the number of edges required to be

f-fault-tolerant can be larger than the number required to be $(f-1)$-fault-tolerant, it is unclear what the dependence on f should be in the approximation ratio. Our third result shows that for $k \in \{3,4\}$ it is possible to remove the dependence on f in the approximation ratio, although we pay for this by only giving bicriteria approximations.

In Chapter 2, we will describe the detailed results which have been published in (Dinitz and Zhang, 2016).

1.2 Distance oracles

Given a finite metric space (V,d), an approximate distance oracle is a data structure which can approximately answer distance queries. It is usually a combination of a preprocessing algorithm to compute a data structure, and a query algorithm which returns a distance $d'(u,v)$ whenever queried on a pair of vertices $u,v \in V$. Similar to the spanners, an approximate distance oracle is said to have multiplicative stretch k if $d(u,v) \leq d'(u,v) \leq k \cdot d(u,v)$.

Note that we can have a trivial stretch 1 distance oracle using $\Theta(n^2)$ space: we could just store the entire metric space. We can also have good space bound but bad query time by storing a spanner. So the primary goal is to to build a small data structure that also has small stretch and small query time.

As the fundamental work in this area, Thorup and Zwick (Thorup and Zwick, 2005) show that for every integer $t \geq 1$, every finite metric space (V,d) has an approximate distance oracle with multiplicative stretch $(2t-1)$ and query time $O(t)$ which uses only $O(tn^{1+\frac{1}{t}})$ space. A significant fraction of more recent results have built off of the ideas developed in (Thorup and

Zwick, 2005), and much of this follow-up work has stored the exact same (or very similar) data structure, just with improved query algorithms or slightly different information in the storage (see, e.g., (Patrascu, Roditty, and Thorup, 2012; Wulff-Nilsen, 2013; Chechik, 2014; Chechik, 2015)).

Most notably, Pătraşcu and Roditty (Patrascu and Roditty, 2010) gave a different distance oracle (although still using some of the basic ideas from (Thorup and Zwick, 2005)) that has multiplicative stretch of 2 and additive stretch of 1 (i.e. $d(u,v) \leq d'(u,v) \leq 2 \cdot d(u,v) + 1$. Note that additive stretch is only meaningful on unweighted graphs), with size $O(n^{\frac{5}{3}})$. This broke through the stretch 3 barrier from (Thorup and Zwick, 2005). Later this result was improved to more general multiplicative/additive stretches (Abraham and Gavoille, 2011).

As we have discussed in the introduction, it is really natural to think about approximating approximate distance oracles, but one of the difficulties is how to make the problem well defined (i.e. what is the meaning of optimizing a data structure given the input instance). In fact, we observed in (Dinitz and Zhang, 2017) that many modern distance oracles (and in particular Thorup-Zwick, Pătraşcu-Roditty, and almost all of their variants) have a similar structure. The preprocessing algorithm chooses to store a subset of the original distances, which has some particular structure. The query algorithm can return a valid distance estimate efficiently as long as the stored distances satisfy the required structure. Thus we can optimize for these particular distance oracles by choosing the best possible set of distances to remember, subject to the required structure. By characterizing this structure for different

types of distance oracles, we can optimize over those types.

For example, the stretch-3 Thorup-Zwick distance oracle uses a subtle but simple method to choose the set of distances to store. It randomly samples a subset of approximately $\sqrt{n}$ vertices, without using any information about the original metric space, and then creates a data structure which is related (in a well-defined, important way) to these vertices. The correctness of the query algorithm does not depend on the choice of the vertices; the choice affects only the size of the data structure. Thus instead of simply choosing the subset of vertices uniformly at random, we can try to optimize the set of chosen vertices with respect to the actual input metric space.

In Chapter 3, we will describe the detailed definitions and results which have been published in (Dinitz and Zhang, 2017).

1.3 Shallow-Light Steiner Network

The SHALLOW-LIGHT STEINER NETWORK problem is defined as follows: Given a graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}^+$, a length function $l : E \to \mathbb{R}^+$, a distance bound L, and p pairs of vertices $\{s_1, t_1\}, \ldots, \{s_p, t_p\}$. The objective of SLSN is to find a minimum cost subgraph $G' = (V, S)$, such that for every $i \in [p]$, there is a path between s_i and t_i in G' with length less or equal to L.

Let H be the graph with vertex set $\{s_1, \ldots, s_p, t_1, \ldots, t_p\}$ and edge set $\{\{s_1, t_1\}, \ldots, \{s_p, t_p\}\}$. We call H the *demand graph* of the problem. We use $|H|$ to represent the number of edges in H. When H is a star (i.e. there is a root $r = s_1 = s_2 = \cdots = s_p$), this is essentially the SHALLOW-LIGHT STEINER

TREE problem.

As we have mentioned, the DIRECTED STEINER NETWORK problem is defined as follows: Given a **directed** graph $G = (V, E)$, an cost function $c : E \to \mathbb{R}^+$, and p pairs of vertices $\{(s_1, t_1), \ldots, (s_p, t_p)\}$. The objective of DSN is to find a minimum cost subgraph $G' = (V, S)$, such that for every $i \in [p]$, there is a path from s_i to t_i in G'.

(Feldmann and Marx, 2016) defined a structure called "almost-caterpillar", which is the union of a constant number of stars where their centers form a path, as well as a constant number of extra edges. Informally, they proved that if the demand graph is transitively equivalent to an "almost-caterpillar", then the DSN problem is in FPT, and otherwise the DSN problem is W[1]-hard.

While *a priori* there might not seem to be much of a relationship between the directed and the length-bounded problems, there are multiple folklore results that relate them, usually by means of some sort of layered graph. For example, any FPT algorithm for the DST problem can be turned into an FPT algorithm for SLST (with unit edge lengths) and vice versa through such a reduction. Such a relationship is not known for more general demands, though.

In light of these relationships between the directed and the length-bounded settings and the recent results of (Feldmann and Marx, 2016), it is natural to attempt to characterize the demand graphs that make SLSN easy or hard. We solve this problem, giving (as in (Feldmann and Marx, 2016)) a complete characterization of easy and hard demand graphs. We show that SLSN is significantly harder than DSN: the only "easy" demand graphs are stars (in

which case the problem is just SLST) and constant-size graphs. Even tiny modifications, like a star with a single independent edge, the problem become W[1]-hard (despite being in FPT for DSN).

In Chapter 4, we will describe the detailed results which have been published in (Babay, Dinitz, and Zhang, 2018).

References

Peleg, David and Alejandro A. Schäffer (1989). "Graph spanners". In: *Journal of Graph Theory* 13.1, pp. 99–116.

Peleg, David and Jeffrey D. Ullman (1989). "An optimal synchronizer for the hypercube". In: *SIAM J. Comput.* 18 (4), pp. 740–747. ISSN: 0097-5397. DOI: http://dx.doi.org/10.1137/0218050. URL: http://dx.doi.org/10.1137/0218050.

Thorup, Mikkel and Uri Zwick (2001). "Compact Routing Schemes". In: *Proceedings of SPAA*. Crete Island, Greece, pp. 1–10. ISBN: 1-58113-409-6. DOI: 10.1145/378580.378581. URL: http://doi.acm.org/10.1145/378580.378581.

Bhattacharyya, Arnab, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff (2009). "Transitive-closure spanners". In: *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 932–941.

Althöfer, Ingo, Gautam Das, David Dobkin, Deborah Joseph, and José Soares (1993). "On sparse spanners of weighted graphs". In: *Discrete Comput. Geom.* 9.1, pp. 81–100. ISSN: 0179-5376. DOI: http://dx.doi.org/10.1007/BF02189308.

Chandra, Barun, Gautam Das, Giri Narasimhan, and José Soares (1992). "New sparseness results on graph spanners". In: *Proceedings of the eighth annual symposium on Computational geometry*. ACM, pp. 192–201.

Kortsarz, G. and D. Peleg (1994). "Generating sparse 2-spanners". In: *J. Algorithms* 17.2, pp. 222–236. ISSN: 0196-6774. DOI: http://dx.doi.org/10.1006/jagm.1994.1032.

Berman, Piotr, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev (2013). "Approximation algorithms for spanner problems and Directed Steiner Forest". In: *Inf. Comput.* 222, pp. 93–107. DOI: 10.1016/j.ic.2012.10.007. URL: http://dx.doi.org/10.1016/j.ic.2012.10.007.

Dinitz, Michael and Robert Krauthgamer (2011a). "Directed spanners via flow-based linear programs". In: *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pp. 323–332. DOI: 10.1145/1993636.1993680. URL: http://doi.acm.org/10.1145/1993636.1993680.

Thorup, Mikkel and Uri Zwick (2005). "Approximate Distance Oracles". In: *J. ACM* 52.1, pp. 1–24. ISSN: 0004-5411. DOI: 10.1145/1044731.1044732. URL: http://doi.acm.org/10.1145/1044731.1044732.

Jacobson, Guy Joseph (1988). "Succinct Static Data Structures". PhD thesis. Pittsburgh, PA, USA: Carnegie Mellon University.

Dreyfus, Stuart E and Robert A Wagner (1971). "The Steiner problem in graphs". In: *Networks* 1.3, pp. 195–207.

Charikar, Moses, Chandra Chekuri, To-yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li (1999). "Approximation algorithms for directed Steiner problems". In: *Journal of Algorithms* 33.1, pp. 73–91.

Zosin, Leonid and Samir Khuller (2002). "On directed Steiner trees". In: *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, pp. 59–63.

Chekuri, Chandra, Guy Even, Anupam Gupta, and Danny Segev (2011). "Set connectivity problems in undirected graphs and the directed steiner network problem". In: *ACM Transactions on Algorithms (TALG)* 7.2, p. 18.

Chlamtác, Eden, Michael Dinitz, Guy Kortsarz, and Bundit Laekhanukit (2017). "Approximating Spanners and Directed Steiner Forest: Upper and Lower Bounds". In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pp. 534–553.

Abboud, Amir and Greg Bodwin (2018). "Reachability Preservers: New Extremal Bounds and Approximation Algorithms". In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pp. 1865–1883.

Feldmann, Andreas Emil and Dániel Marx (2016). "The Complexity Landscape of Fixed-Parameter Directed Steiner Network Problems". In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*. Vol. 55. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 27:1–27:14.

Dinitz, Michael and Zeyu Zhang (2016). "Approximating low-stretch spanners". In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, pp. 821–840.

Dinitz, Michael, Guy Kortsarz, and Ran Raz (2012). "Label Cover Instances with Large Girth and the Hardness of Approximating Basic k-Spanner". In: *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP)*. DOI: 10.1007/978-3-642-31594-7_25. URL: http://dx.doi.org/10.1007/978-3-642-31594-7_25.

Levcopoulos, Christos, Giri Narasimhan, and Michiel Smid (1998). "Efficient algorithms for constructing fault-tolerant geometric spanners". In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, pp. 186–195.

Chechik, Shiri, Michael Langberg, David Peleg, and Liam Roditty (2010). "Fault Tolerant Spanners for General Graphs". In: *SIAM J. Comput.* 39.7, pp. 3403–3423. DOI: 10.1137/090758039. URL: http://dx.doi.org/10.1137/090758039.

Dinitz, Michael and Robert Krauthgamer (2011b). "Fault-tolerant spanners: better and simpler". In: *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. DOI: 10.1145/1993806.1993830. URL: http://doi.acm.org/10.1145/1993806.1993830.

Patrascu, Mihai, Liam Roditty, and Mikkel Thorup (2012). "A new infinity of distance oracles for sparse graphs". In: *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, pp. 738–747.

Wulff-Nilsen, Christian (2013). "Approximate distance oracles with improved query time". In: *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, pp. 539–549.

Chechik, Shiri (2014). "Approximate distance oracles with constant query time". In: *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*. ACM, pp. 654–663.

Chechik, Shiri (2015). "Approximate distance oracles with improved bounds". In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*. ACM, pp. 1–10.

Patrascu, Mihai and Liam Roditty (2010). "Distance oracles beyond the Thorup-Zwick bound". In: *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, pp. 815–823.

Abraham, Ittai and Cyril Gavoille (2011). "On approximate distance labels and routing schemes with affine stretch". In: *Proceedings of the International Symposium on Distributed Computing (DISC)*. Springer, pp. 404–415.

Dinitz, Michael and Zeyu Zhang (2017). "Approximating Approximate Distance Oracles". In: *Proceedings of the 8th Innovations in Theoretical Computer Science conference*. Vol. 23.

Babay, Amy, Michael Dinitz, and Zeyu Zhang (2018). "Characterizing demand graphs for (fixed-parameter) shallow-light steiner network". In: *Proceedings of the 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*. Vol. 33.

Chapter 2

Spanners

2.1 Definitions and Preliminaries

Given a graph $G = (V, E)$, the distance $d_G(u, v)$ between any two nodes u, v is the length of the shortest path between them. While in many contexts it is interesting to allow arbitrary edge lengths, in this thesis we consider the more basic setting of unit-length spanners (i.e., the distance between two nodes is the number of hops in a shortest path). When the graph is clear from context, we will sometimes drop the subscript and refer to $d(u, v)$. We will typically let $n = |V(G)|$. For any subset $S \subseteq V$, we will let $G \setminus S$ denote the subgraph of G induced by $V \setminus S$.

A subgraph H of G is a k-spanner of G if $d_H(u, v) \leq k \cdot d_G(u, v)$ for all $u, v \in V$. The value k is referred to as the *stretch*. Note that it is necessary and sufficient for $d_H(u, v) \leq k$ for all edges $\{u, v\} \in E$, since if every edge has its distance (approximately) preserved then all pairs have their distances (approximately) preserved. Hence we will typically prove that all edges have stretch at most k. In the BASIC k-SPANNER problem, we are given a graph

G and an integer $k \geq 2$. The goal is to output a k-spanner of G with as few edges as possible. DIRECTED k-SPANNER is the same problem but where G is directed.

Chechik et al (Chechik et al., 2010) extended the definition of a spanner to an f-fault-tolerant k-spanner: a subgraph H of G is an f-fault-tolerant k-spanner if $d_{H \setminus F}(u, v) \leq k \cdot d_{G \setminus F}(u, v)$ for all $u, v \in V$ and $F \subseteq V$ with $|F| \leq f$. In other words, $H \setminus F$ must be a k-spanner of $G \setminus F$ for all sets $F \subseteq V$ with size at most f. In the f-FAULT-TOLERANT k-SPANNER problem, we are given a graph G and integers k, f. The goal is to output an f-fault-tolerant k-spanner of G with as few edges as possible.

2.1.1 Results and Techniques

For BASIC k-SPANNER with $k \geq 3$, while (Althöfer et al., 1993) provided a trivial upper bound of $n^{2/k}$ (even k) or $n^{2/(k+1)}$ (odd k) on the approximation ratio, the only case in which we know how to beat this bound is for $k = 3$: an $\tilde{O}(n^{1/3})$-approximation was given by (Berman et al., 2013). Moreover, for $k \geq 5$ the trivial bound is already at most $n^{1/3}$, so in fact $k = 4$ is the only stretch value for which an $\tilde{O}(n^{1/3})$-approximation was not known. Our first result closes this gap, proving the following theorem.

Theorem 2.1.1. *There is a $\tilde{O}(n^{1/3})$-approximation for* BASIC 4-SPANNER *(and for* DIRECTED 4-SPANNER*).*

The main innovation of this algorithm is a new LP rounding algorithm for the standard flow-based LP which generalizes both of the previous algorithms for the $k = 3$ case (Dinitz and Krauthgamer, 2011a; Berman et al., 2013). This

rounding algorithm picks any edge that (Dinitz and Krauthgamer, 2011a) or (Berman et al., 2013) would have picked, but also includes additional edges without increasing the cost by more than an $O(\log n)$ factor. The analysis is a combination of careful bucketing, allowing us to assume that a significant amount of flow uses paths with similar structure which then form subgraphs which are extremely regular, and case analysis.

After that, we consider the approximation lower bound. While (Dinitz, Kortsarz, and Raz, 2012) proved a superpolylogarithmic lower bound for BASIC k-SPANNER, there is still a significant gap between the lower bound of (Dinitz, Kortsarz, and Raz, 2012) and the trivial upper bound of (Althöfer et al., 1993). Even assuming polynomial hardness for LABEL COVER, it remains possible that there is (for example) an $n^{1/(100k)}$-approximation algorithm for BASIC k-SPANNER, which would be an enormous improvement over the upper bound from (Althöfer et al., 1993), particularly for small k. Our second result provides some evidence that the upper bound is essentially tight (or at least, if improvements are possible then they must use stronger relaxations).

Theorem 2.1.2. *Assuming the Erdős girth conjecture, for any constant $\delta > 0$, the integrality gap of the flow LP for* BASIC k-SPANNER *is at least $\Omega(n^{\frac{2}{(1+\delta)(k+1)+4}})$ for odd k or $\Omega(n^{\frac{2}{(1+\delta)k+4}})$ for even k.*

In other words, the dependence on k of the LP is (up to an additive 4 in the denominator) essentially the same as the upper bound from (Althöfer et al., 1993). This is particularly notable since this flow LP (or its equivalents) is the basic building block for essentially all known spanner approximations other than (Althöfer et al., 1993), including the best known algorithm for BASIC

3-SPANNER (Berman et al., 2013).

The Erdős girth conjecture (given here as Conjecture 2.4.1) is a longstanding open question about the maximum density possible in large girth graphs. If this conjecture turns out to be false, then Theorem 2.1.2 can be replaced by a theorem in which k is replaced by some value that is a function the degrees of these extreme graphs. While this would be weaker, the upper bound of (Althöfer et al., 1993) would immediately be stronger, and once again the integrality gap would essentially match the upper bound.

At a high level, this integrality gap is similar to the $\Omega(n^{1/3})$ integrality gap from (Dinitz and Krauthgamer, 2011a) for the same LP in the directed setting: the hardness reduction of (Elkin and Peleg, 2000; Kortsarz, 2001) is applied not to hard instances of LABEL COVER (or its minimization version, MIN-REP), but to *random* instances. This forces the integral optimum to be large, but small fractional solutions still exist. In order to make this work in the undirected context, we need to use the ideas of (Dinitz, Kortsarz, and Raz, 2012) rather than the simple hardness reduction of (Elkin and Peleg, 2000). In particular, we use a random instance of MIN-REP which have some additional properties. It contains a fixed dense supergraph with large supergirth from Erdős girth conjecture, but the connections between partitions are random. This requires a new analysis of random instances of LABEL COVER, which together with some parameter-tuning is enough to give the result.

We then turn our attention to low-stretch fault-tolerant spanners. Our goal is to study the dependency on f, and in particular to remove all such

dependency from the approximation. The best known results depend polynomially on f: for any $k \geq 3$ there is an $\tilde{O}(fn^{1/\lfloor (k+1)/2 \rfloor})$-approximation (this is straightforward from the absolute bound of (Dinitz and Krauthgamer, 2011b)). For directed graphs the situation is far worse: the best dependence known on f is *exponential*, except for $k = 3$ where the dependence is linear (Dinitz and Krauthgamer, 2011a).

By using a new LP relaxation (modeled after the improved relaxation for f-FAULT-TOLERANT 2-SPANNER from (Dinitz and Krauthgamer, 2011b)) we are able to remove all dependence on f when $k \in \{3, 4\}$, but at the price of a bicriteria approximation. An algorithm is an (α, β)-approximation for f-FAULT-TOLERANT k-SPANNER if on all inputs, it returns an αf-fault-tolerant k-spanner with at most $\beta \cdot OPT$ edges, where OPT is the size of the sparsest f-fault-tolerant k-spanner.

Theorem 2.1.3. *For $k \in \{3, 4\}$, $f \leq O(\sqrt{n})$, and any arbitrarily small constant $\epsilon > 0$, there is a $(1 - \varepsilon, \tilde{O}(\sqrt{n}))$-approximation algorithm for f-FAULT-TOLERANT k-SPANNER and for f-FAULT-TOLERANT DIRECTED k-SPANNER.*

Proving this theorem requires extending Theorem 2.1.1 in two ways: using a new LP relaxation, and a more sensitive analysis in which the high probability guarantees are high enough that we can do a union bound over all fault sets. It straightforward to achieve this kind of high probability guarantee by losing factors of f in the rounding, but obtaining it without losing factors of f turns out to be extremely challenging.

2.1.2 Outline

We begin in Section 2.1.3 by defining the natural LP, which will be used in our algorithms and for which we will prove the integrality gap. In Sections 2.2 and 2.3 we discuss our new algorithm for directed and basic 4-spanner, where Section 2.2 gives the algorithm and a general overview of the techniques while Section 2.3 gives the formal proofs. We then move to the integrality gap for BASIC k-SPANNER, giving the basic ideas in Section 2.4 and the detailed proofs in Section 2.5. Finally, we give our algorithm for f-FAULT-TOLERANT k-SPANNER (for $k = 3, 4$) and an extremely high-level view of the analysis in Section 2.6, with details in Section 2.7

2.1.3 The flow-based LP for BASIC k-SPANNER

The flow LP for BASIC k-SPANNER (which was initially introduced by (Dinitz and Krauthgamer, 2011a) for the directed version) is given in Figure 2.1, where $\mathcal{P}_{u,v}$ denotes the collection of all $u - v$ paths of length at most k. Intuitively, one can think of the x variables as capacities and the y variables as flow values, and the LP is set up to minimize total capacity (integrally, number of edges included) subject to being able to send at least 1 unit of flow from u to v along stretch k paths for all $\{u, v\} \in E$ (integrally, some stretch-k path must exist). This is also clearly a valid relaxation for the directed setting.

We note that this LP can clearly be solved in polynomial time for constant k, which will be useful when designing our algorithm in Section 2.2. When k is superconstant, it can be approximately solved to within a $(1 + \epsilon)$-factor of the optimal solution (Dinitz and Krauthgamer, 2011a).

$$
\begin{array}{lll}
\min & \sum_{e \in E} x_e & \\
\text{s.t.} & \sum_{P \in \mathcal{P}_{u,v}: e \in P} f_P \leq x_e & \forall \{u,v\} \in E,\ \forall e \in E \\
& \sum_{P \in \mathcal{P}_{u,v}} f_P \geq 1 & \forall \{u,v\} \in E \\
& x_e \geq 0 & \forall e \in E \\
& f_P \geq 0 & \forall \{u,v\} \in E,\ \forall P \in \mathcal{P}_{u,v}
\end{array}
\tag{2.1}
$$

Figure 2.1: LP relaxation for BASIC k-SPANNER

2.2 Directed and Basic 4-Spanner

We describe our algorithm in the directed setting, but it is straightforward to see that it gives the same bounds in the undirected setting. In this section we give the algorithm and an informal overview of the proof. The full proof follows in Section 2.3.

2.2.1 Algorithm

Our algorithm follows (Dinitz and Krauthgamer, 2011a; Berman et al., 2013) in having two parts: a tree-sampling algorithm to handle "thick" edges, and an LP-rounding algorithm to handle "thin" edges. We use the basic flow LP (2.1) with $k = 4$. Define the *local neighborhood* of (u, v) to be $V_{u,v} = \{w \in V : w \in P \text{ for some } P \in \mathcal{P}_{u,v}\}$, the set of nodes that are in at least one spanning path for (u, v). For a parameter $\beta = n^{1/3}$, we say that (u, v) is *thin* if $|V_{u,v}| \leq n/\beta$ and that otherwise it is *thick*. We say that a subset of edges E' *settles* (u, v) if there is some $P \in \mathcal{P}_{u,v}$ such that the edges of P are contained in E'. Thus the optimal 4-spanner H is exactly the smallest edge set which settles all $(u, v) \in E(G)$.

Thick edges can be handled in the same way as in (Dinitz and Krauthgamer,

2011a) and (Berman et al., 2013), through random arborescence sampling. The following theorem is implicit in (Dinitz and Krauthgamer, 2011a) and explicit in (Berman et al., 2013).[1]

Theorem 2.2.1. *There is a randomized algorithm to construct a set of edges E_1 such that E_1 settles all thick edges (u, v). Moreover, the expected size of E_1 is at most $3\beta \ln n \cdot OPT$.*

It remains to handle thin edges. For this we will first solve the LP relaxation, and then round the produced fractional solution $\vec{x}$ to get a set of edges E_2. Our algorithm is given formally in Algorithm 1. Our rounding combines the Poisson sampling technique of (Berman et al., 2013) with a generalization of the "square-root rounding" of (Dinitz and Krauthgamer, 2011a).

Informally, we first use the basic Poisson sampling step of (Berman et al., 2013) to ensure that all fractional values are multiples of β/n without losing more than a constant factor in the cost and while maintaining feasibility. We then have each vertex u draw uniformly at random a value r_u from $[0, 1]$. We then add each edge $(u, v) \in G$ to E_2 if $r_u r_v \leq \alpha x_e = \tilde{O}(\beta x_e)$; in other words, we include e if the product of the values of its endpoints is at most the LP value of the edge (inflated by β). The final set of edges returned by our algorithm is $E_1 \cup E_2$.

This generalizes previous algorithms for related problems (DIRECTED 3-SPANNER in particular): Berman et al. (Berman et al., 2013) include (u, v) if either r_u or r_v is at most $\tilde{O}(\beta x_e)$, and Dinitz and Krauthgamer (Dinitz

[1] We note that defining some type of "local neighborhood", using arborescence sampling in some cases, and LP rounding in others, was used earlier (with different terminology) in the context of the DIRECTED STEINER FOREST problem (Feldman, Kortsarz, and Nutov, 2012).

Algorithm 1 Rounding algorithm for thin edges

$E_2 \leftarrow \varnothing$
$C_1 \leftarrow 600, C_2 \leftarrow 8000$
$\alpha \leftarrow C_2 \beta \ln^6 n$
for each edge $e \in E$ **do**
    $p_e \leftarrow$ sample from the Poisson distribution with parameter $\lambda_e = C_1 n x_e / \beta$
    $\hat{x}_e \leftarrow \beta p_e / n$
end for
for each vertex $v \in V$ **do**
    $r_v \leftarrow$ uniform sample from $[0, 1]$
end for
for each edge $e = (s, t) \in E$ **do**
    if $r_s \cdot r_t \leq \alpha \hat{x}_e$ **then**
        add e to E_2
    end if
end for
return E_2

and Krauthgamer, 2011a) include x_e if both r_u and r_v are at most $\tilde{O}(\sqrt{\beta x_e})$. Informally, Berman et al. required one endpoint to "buy" the edge, while Dinitz and Krauthgamer require both endpoints to evenly split the cost of the edge. We, on the other hand, allow the two endpoints to together buy the edge using uneven splits.

This new "product rule" is, despite its simplicity and obvious intuition, the major algorithmic change from (Berman et al., 2013) and (Dinitz and Krauthgamer, 2011a). It is not difficult to construct examples where both (Berman et al., 2013) and (Dinitz and Krauthgamer, 2011a) fail when $k = 4$, but our generalized algorithm succeeds. Moreover, it gives us enormous flexibility in the analysis. Depending on where in the proof we are and what case we are considering, we can define thresholds p_u and p_v and say that

we "select" edge (u, v) if $r_u \leq p_u$ and $r_v \leq p_v$. As long as $p_u p_v \leq \alpha \hat{x}_e$, any edge we select is actually chosen by the algorithm. The algorithms of (Berman et al., 2013) and (Dinitz and Krauthgamer, 2011a) can both be thought of as hardcoding p_u and p_v into the algorithm, while we can use different threshold values in different cases. This turns out to make all the difference.

We note that Algorithm 1 does "work" for all edges (not just thin edges) if β is replaced by $\Theta(\sqrt{n})$, but this would only give an $\tilde{O}(\sqrt{n})$-approximation. This is essentially how (Dinitz and Krauthgamer, 2011a) gave an $\tilde{O}(\sqrt{n})$-approximation when $k = 3$. But the performance of Algorithm 1 degrades when local neighborhoods get large, while the performance of the algorithm in Theorem 2.2.1 improves when local neighborhoods are large. By trading them off appropriately, we get our bound of $\tilde{O}(n^{1/3})$.

2.2.2 Overview of Analysis

Cost:

We defer the proof of the cost, as well as other detailed proofs in this section, to section 2.3, and just state the following theorem.

Theorem 2.2.2. $\mathbb{E}[|E_1 \cup E_2|] \leq \tilde{O}(n^{1/3} \cdot OPT)$.

Correctness:

We now focus on proving the following theorem, which together with Theorem 2.2.2 implies Theorem 2.1.1.

Theorem 2.2.3. *$E_1 \cup E_2$ is a 4-spanner of G with probability at least $1 - \frac{2}{n}$.*

In other words, we need to show that with high probability, every edge

is settled by $E_1 \cup E_2$. This is true for the thick edges by Theorem 2.2.1, so we need to prove this only for thin edges.

Let $(s,t) \in E$ be a thin edge. We claim that the probability that E_2 settles (s,t) is at least $1 - \frac{1}{n^3}$, which then by a union bound implies Theorem 2.2.3. It is relatively straightforward to show that the Poisson rounding, when we changed edge capacities from x_e to $\hat{x}_e$, does not have much impact on the flow (see Lemma 2.3.7 for the formal statement). So we will simply assume that after Poisson rounding we still have essentially the same amount of flow, and ignore the Poisson rounding for the rest of this section (it is a technical detail which is important in some cases in the main proof, as it allows us to lower bound the minimum capacity on any edge which has nonzero flow, but we will not go into details here).

It is easy to see that if a constant amount of flow is sent from s to t along paths of length 1, 2, or 3 then E_2 settles (s,t) with high probability. This is because the algorithm of (Berman et al., 2013) suffices for those cases, and any edge picked by (Berman et al., 2013) is picked by our algorithm. So without loss of generality a constant fraction of the flow is sent along paths of length exactly 4. Again without loss of generality, by using standard partitioning tricks we can assume there is a partition S_1, S_2, S_3 of $V_{s,t} \setminus \{s,t\}$ such that at least a constant amount of flow is on paths of the form $s \to S_1 \to S_2 \to S_3 \to t$. Let $\mathcal{P}'$ denote this set of paths and let $E^*_{s,t}$ denote the edges which are used by at least one path in $\mathcal{P}'$. For each edge $e \in E^*_{s,t}$, let f_e denote the total amount of flow along paths in $\mathcal{P}'$ which use edge e.

So now we just need to show that E_2 contains a path from $\mathcal{P}'$ with high

probability. For each $v \in V_{s,t}$, define the *load* l_v to be $l_v = \sum_{P \in \mathcal{P}' : v \in P} f_P$, i.e., the total flow on paths in $\mathcal{P}'$ which passes through v.

If there is a node $v \in S_2$ with $l_v \geq 1/\beta$, then it turns out to be relatively easy to prove that E_2 settles (s,t) with high probability. Intuitively, this is because there is so much flow going through v that with high probability we choose a path of length 2 from s to v and with high probability we choose a path of length 2 from v to t.

The more difficult case is when no node in S_2 has large load, i.e., the flow from s to t is very "spread out". For each $P \in \mathcal{P}'$, let u_P denote the S_1 node in P, let v_P denote the S_2 node in P, and let w_P denote the S_3 node in P. We first bucket the paths in $\mathcal{P}'$ based on the load of the S_1 node (l_{u_P}), the load of the S_3 node (l_{w_P}), and the amount of flow in total along the second and third hops ($f_{(u_P,v_P)}$ and $f_{(v_P,w_P)}$) and pick the bucket with the largest amount of flow. This costs us a polylogarithmic factor in the amount of flow, but now all of the paths that we consider have the same values for these parameters (up to a factor of 2 in each parameter). We now split into four more cases depending on how these parameters relate to each other.

The first case (and the easiest) is if l_{u_P} and l_{w_P} are both at least $1/\alpha$. In this case we get the first and last hop of each path with probability 1, making it straightforward to argue that there is a high probability of getting an entire path. The tricker cases are when we do not have large loads at both u_P and w_P. There are three cases, depending on whether both of them have small load or whether one (or the other) has small load. All of these cases work in basically the same way, but with different parameters. In particular, we will

use an “iterative sampling" method for all of them. This method will also turn out to be useful in the fault-tolerant setting, so we state the corresponding lemma in full generality and include a brief discussion.

Iterative Sampling:

Let $G = (L_1, \ldots, L_l, E)$ be a layered graph with l layers and at most n nodes (by layered, we mean that all edges are between adjacent layers). Let $\mathcal{P}$ be the set of all paths of the form $L_1 \to L_2 \to \cdots \to L_l$. For each $P \in \mathcal{P}$, let f_P be the flow along this path. For each $i \in [l]$ and each node $v \in L_i$, suppose that $\sum_{P \in \mathcal{P}: v \in P} f_P \leq c_i$ (so the load is at most c_i for each vertex in L_i). Let $p_i \in \mathbb{R}^{\geq 0}$ be a threshold value for each layer i. For each vertex v, let r_v be a value drawn independently and uniformly at random from $[0, 1]$, and say that $v \in L_i$ is “good" if $r_v \leq p_i$. We call a path *j-good* if it goes from the first layer to the j-th layer and all nodes in the path are good.

Lemma 2.2.4. *If*

$$\frac{\sum_{P \in \mathcal{P}} f_P}{t} > \left\lceil \sum_{i=1}^{l} \left(c_i \cdot \prod_{j=i}^{l} \left\lceil \frac{4 \ln n}{p_j} \right\rceil \right) \right\rceil$$

for an integer t, then with probability at least $1 - \frac{1}{n^3}$, there are t node-disjoint l-good paths.

The intuition behind Lemma 2.2.4 is that if the total flow is large compared to the node capacities (i.e. the loads) then the flow must be very spread out, so it should be possible to find disjoint paths (for this particular case multiple paths are not necessary, but they will become useful when considering the fault-tolerant setting). The proof of Lemma 2.2.4 works by flipping the random coins in a very specific order (since all r_v values are independent we can flip

them in any order that we like). We first flip only some of the coins for L_1, and then analyze how much flow is "used up" and how many nodes in L_2 are adjacent to a good node in L_1. We keep repeating this throughout the layers, and by doing this carefully can show that we find such a path with high probability. It then turns out that if the flow is large enough there are still unflipped coins left, so we can repeat using only nodes that we did not already consider. Again, if the flow is large enough we can keep repeating until we have our disjoint paths. We call this technique "iterative sampling" due to the iterative nature of the coin flips.

Use in 4-Spanner:

As an example of how Lemma 2.2.4 is used, suppose we are in the case where both l_{u_P} and l_{w_P} are less than $1/\alpha$. Then we can prove that every edge of P is included if we have $r_{u_P} \leq \alpha l_{u_P}$ and $r_{w_P} \leq \alpha l_{w_P}$ and $r_{v_P} \leq \min\{f_{(u_P,v_P)}/l_{u_P}, f_{(v_P,w_P)}/l_{w_P}\}$. So by thinking of L_i as S_i for $i \in \{1,2,3\}$ and each of the above thresholds as the thresholds p_i, we can apply Lemma 2.2.4 with $t = 1$ to get a path from s to t with high probability. The other two cases work the same way, just with different sufficient conditions for a path to be included. This is an example of where the flexibility of the product rule is extremely helpful, as we can set different thresholds for the different cases.

2.3 Complete Analysis for 4-Spanner

We begin by formally analyzing the cost of the algorithm, before proving that it does indeed compute a 4-spanner with high probability.

2.3.1 Cost of Algorithm 1: Proof of Theorem 2.2.2

In order to prove Theorem 2.2.2, we first need an easy lemma about the Poisson distribution. This lemma can be found as Lemma A.1 in (Berman et al., 2013).

Lemma 2.3.1. *Let X be a Poisson random variable with parameter $\lambda \geq 1$. Then* $\mathbf{Pr}\left[X < \left\lfloor \frac{\lambda}{e} \right\rfloor\right] \leq e^{-\frac{\lambda}{4}}$ *and* $\mathbf{Pr}[X > e\lambda] \leq e^{-\lambda}$.

We analyze the cost of the Poisson rounding process in the same way as in (Berman et al., 2013) but with different parameters, to get the following lemma. This lemma shows that with very high probability, the total weight after Poisson rounding will not be much larger than the total weight of the original LP solution.

Lemma 2.3.2. $\mathbf{Pr}\left[\sum_{e\in E} \hat{x}_e > eC_1 \sum_{e\in E} x_e\right] < e^{-C_1 \frac{n}{\beta}}$.

Proof. Since the summation of Poisson distribution is still a Poisson distribution, we have that $\sum_{e\in E} p_e$ is a Poisson random variable with parameter $\sum_{e\in E} \lambda_e$. Hence by Lemma 2.3.1 we get that

$$\begin{aligned}
\mathbf{Pr}\left[\sum_{e\in E} \hat{x}_e \geq e \cdot C_1 \sum_{e\in E} x_e\right] &= \mathbf{Pr}\left[\sum_{e\in E} \frac{\beta}{n} p_e \geq e \cdot C_1 \sum_{e\in E} x_e\right] \\
&= \mathbf{Pr}\left[\sum_{e\in E} p_e \geq e \sum_{e\in E} \lambda_e\right] \\
&\leq e^{-\sum_{e\in E} \lambda_e} = e^{-C_1 \frac{n}{\beta} \sum_{e\in E} x_e} < e^{-C_1 \frac{n}{\beta}},
\end{aligned}$$

proving the lemma. □

We can now calculate the expected cost of our product rule in terms of the $\hat{x}$ variables.

Lemma 2.3.3. $\mathbb{E}[|E_2|] \leq O(\alpha \ln n \cdot \sum_{\{u,v\} \in E} \hat{x}_{u,v})$

Proof. Let $X_{u,v}$ be the indicator variable for the event that $e = (u, v)$ is included in E_2. We claim that $\mathbf{Pr}[X_{u,v} = 1] \leq O(\alpha \ln n \cdot \hat{x}_{u,v})$. This clearly implies the theorem by linearity of expectations. Note that the claim is trivially true if $\hat{x}_e = 0$, since in that case the probability that the algorithm selects (u, v) is 0. The claim is also true if $\alpha \hat{x}_e \geq 1$. So we will assume that $\hat{x}_e > 0$ and $\alpha \hat{x}_e < 1$, which by the Poisson rounding implies that $\frac{\beta}{n} \leq \hat{x}_e \leq \frac{1}{\alpha}$. Note that once r_v has been chosen, the probability that (u, v) is added to E_2 is $\min\{1, \frac{\alpha \hat{x}_{u,v}}{r_v}\}$. Hence the probability that (u, v) is included in H is

$$\begin{aligned}
\int_0^1 \min\left(1, \frac{\alpha \hat{x}_{u,v}}{r_v}\right) dr_v &= \int_0^{\alpha \hat{x}_{u,v}} 1 dr_v + \int_{\alpha \hat{x}_{u,v}}^1 \frac{\alpha \hat{x}_{u,v}}{r_v} dr_v \\
&= \alpha \hat{x}_{u,v} + \alpha \hat{x}_{u,v} (\ln 1 - \ln \alpha \hat{x}_{u,v}) \\
&= \alpha \hat{x}_{u,v} \left(1 + \ln \frac{1}{\alpha \hat{x}_{u,v}}\right) \\
&\leq O(\ln n) \cdot \alpha \hat{x}_{u,v},
\end{aligned}$$

where in the final step we used that $\alpha \hat{x}_{u,v} \geq \hat{x}_{u,v} \geq \frac{\beta}{n} \geq \frac{1}{n}$. Hence $\mathbf{Pr}[X_{u,v} = 1] \leq O(\alpha \ln n \cdot \hat{x}_{u,v})$. □

Now we can bound the actual cost of $E_1 \cup E_2$, the subgraph that our algorithm returns.

Note that Theorem 2.2.1 directly implies that $\mathbb{E}[|E_1|] \leq \tilde{O}(\beta \cdot OPT)$. To analyze E_2, by combining Lemma 2.3.3 with Lemma 2.3.2 and $\alpha = C_2 \beta \ln^6 n$

we get that

$$\begin{aligned}
\mathbb{E}[|E_2|] &\leq \tilde{O}\left(\beta \sum_{e \in E} \hat{x}_e\right) \\
&\leq \tilde{O}\left(\beta\left(e^{-C_1 \frac{n}{\beta}} n^2 + eC_1 \sum_{e \in E} x_e\right)\right) \\
&\leq \tilde{O}\left(\beta \sum_{e \in E} x_e\right) \\
&\leq \tilde{O}(\beta \cdot OPT).
\end{aligned}$$

The second inequality is because when we fail to control the summation of $\hat{x}_e$, the total number of edges is at most n^2.

Linearity of expectations, and the fact that $\beta = n^{1/3}$, implies the Theorem 2.2.2

2.3.2 Correctness of Algorithm 1

In order to prove Theorem 2.2.3, we need a number of useful lemmas. We first give a general rounding technique we call “iterative sampling" (discussed earlier in Section 2.2.2), which will allow us to handle many cases in the main proof and will also be useful in the fault-tolerant setting. We then show (in Lemma 2.3.7) that after Poisson rounding, we can find a layered graph where the max flow is still large. Finally we move to the heart of the proof, Lemma 2.3.8, which shows that if a significant amount of flow is sent along paths of length 4 then E_2 settles (s, t) with high probability.

2.3.2.1 Iterative Sampling: Proof of Lemma 2.2.4

We begin with the proof of Lemma 2.2.4. This is an extremely general lemma which will also be useful in the fault-tolerant setting, so we prove it in full generality. Note, however, that for the purpose of proving Theorem 2.2.3 it is overkill – we will only ever need to set $t = 1$.

Recall the basic definitions for this lemma. Let $G = (L_1, \ldots, L_l, E)$ be a layered graph with l layers and at most n nodes (by layered, we mean that all edges are between adjacent layers). Let $\mathcal{P}$ be all the possible paths from L_1 to L_l of length $l - 1$ (i.e., paths of the form $L_1 \to L_2 \to \cdots \to L_l$). For each $P \in \mathcal{P}$, let f_P be the flow along this path. For each $i \in [l]$ and each node $v \in L_i$, suppose that $\sum_{P \in \mathcal{P}: v \in P} f_P \leq c_i$ (so the "load" is at most c_i for each vertex in L_i). Let $p_i \in \mathbb{R}^{\geq 0}$ be a threshold value for each layer i. For each vertex v, let r_v be a value drawn independently and uniformly at random from $[0, 1]$, and say that $v \in L_i$ is "good" if $r_v \leq p_i$. We call a path *j-good* if it goes from the first layer to the j-th layer and all nodes in the path are good.

For any vertex set S, we define $N(S)$ as the neighbor of set S. Because all the r_v are assigned independently, we can arbitrarily choose the sampling order in order to make our analysis easier. We just need to be careful that we never choose r_v for the same node v more than once.

Consider the recursive procedure $\textsc{Sample}(j, R)$ as follows. We will show that, if we run $\textsc{Sample}(j, \varnothing)$ successfully, it will return an endpoint of a j-good path and hence we successfully sampled a j-good path. In order to get more than one path, we will run $\textsc{Sample}$ t times. But instead of initializing $R = \varnothing$, on the t'th iteration we will set it to the final value of R at the end of the

$(t-1)$'st iteration. In other words, R is essentially a global variable which keeps track of which nodes have already sampled their r_v value. We will show that with high probability (at least $1 - \frac{1}{n^3}$) we can run SAMPLE successfully t times, which suffices to prove the lemma.

We emphasize that while SAMPLE is phrased as an algorithm, it is an analytical and not algorithmic tool. Lemma 2.2.4 is purely structural, and we do not actually use SAMPLE as part of any algorithm.

Algorithm 2 Iterative sampling

procedure SAMPLE(j, R)
  $S_j \leftarrow \varnothing$
  for $i \in \left[\left\lceil \frac{4 \ln n}{p_j} \right\rceil\right]$ **do**
    if $j = 1$ **then**
      $v \leftarrow$ arbitrary node in $L_1 \backslash R$
    else
      $(ans, R) \leftarrow$SAMPLE$(j-1, R)$
      $v \leftarrow$ arbitrary node in $L_j \cap N(ans) \backslash R$
    end if
    //add v to S, S is always a subset of $L_j \backslash R$
    $S \leftarrow S \cup \{v\}$
    //sample v
    $r_v \leftarrow$ uniform sample from $[0, 1]$
    //remove the sampled node
    $R \leftarrow R \cup \{v\}$
    //remove nodes which no longer have any flow going through after removing R
    while $\exists u \in L_k \backslash R, k \in [j-1]$ such that $\sum_{P \in \mathcal{P}: u \in P, P \cap R = \varnothing} f_P = 0$ **do**
      $R \leftarrow R \cup \{u\}$
    end while
  end for
  //S' is the good nodes in S
  $S' \leftarrow \{v \mid v \in S, r_v \leq p_j\}$
  $ans \leftarrow$ arbitrary node in S'
  return ans, R
end procedure

First, note that a trivial induction implies that throughout the algorithm, R contains the set of nodes where r_v has already been chosen. Hence no node will have its r_v value sampled more than once.

This procedure first recursively picks the endpoints of many disjoint $(j-1)$-good paths. Because we sample many $(j-1)$-good paths, with high probability we can extend at least one of them to a j-good path. We begin by showing that if the algorithm does not get stuck due to some set that it is trying to select from being empty, then the algorithm successfully finds a path.

Claim 2.3.4. *If we can always perform line 5, line 8 and line 23 while recursively calling them in this procedure (i.e. these sets are always non-empty), then* SAMPLE(j, R) *can find an endpoint of j-good path.*

Proof. We do induction on j. In the base case, if $j = 1$ then if we can always perform line 5 and line 23, *ans* itself is a 1-good path because all nodes in S' are good.

For the inductive case, assume the claim is true for $j = k$ and consider $j = k + 1$. In each iteration i, the procedure runs SAMPLE(k, R) and gets the endpoint of a k-good path (by induction), and finds its neighbor v in $L_{k+1} \backslash R$ to extend this path to $(k+1)$-st layer (such a vertex exists by the assumption that we can execute line 8). Then we sample v, move it to the sampled set R, and clean-up all the useless nodes (i.e., remove all nodes u where all of the remaining flow for u went through v). After that, the next iteration will generate another k-good path. This path must be disjoint from the former ones, because every node in the former paths must have been put into R and will never be selected again. Also, this path can be extended to layer $(k+1)$

because line 8 can always be performed.

When we have finished all the iterations, we get a endpoint set S in the $(k+1)$-st layer of these disjoint paths. All nodes in these paths are good except (possibly) the nodes in S. However, if we can always perform line 23, it means that $ans \in S$ is good, which means the procedure finds a $(k+1)$-good path. □

Claim 2.3.5. *While $\sum_{P\in\mathcal{P}:P\cap R=\varnothing} f_P > 0$, we can always perform line 5 and line 8.*

Proof. If $\sum_{P\in\mathcal{P}:P\cap R=\varnothing} f_P > 0$, then $L_1\backslash R \neq \varnothing$ because there must be some nodes in L_1 such that these flow end with. So we can always choose a node in L_1 in line 5.

Also, in procedure SAMPLE$(k+1)$, any possible return of SAMPLE(k, R) (i.e. any node in S_k) must have a neighbor in $L_{k+1}\backslash R$. This is because each time before we do SAMPLE(k, R), every node in $L_k\backslash R$ must not be useless (i.e. there is some flow pass through it, and all the nodes in the flow are not in R), so it connects to some node in $L_{k+1}\backslash R$. Hence we can always perform line 8. □

Claim 2.3.6. *Suppose that we run* SAMPLE(l, R) *t times, but instead of always initializing $R = \varnothing$, on the t'th iteration we will set it to the final value of R at the end of the $(t-1)$'st iteration. Then $\sum_{P\in\mathcal{P}:P\cap R=\varnothing} f_P$ will always be greater than 0, and with probability at least $1 - \frac{1}{n^3}$, we can perform line 23 on every call.*

Proof. It is easy to see that the total remaining flow $\sum_{P\in\mathcal{P}:P\cap R=\varnothing} f_P$ decreases only when a useful node (i.e. a node with $\sum_{P\in\mathcal{P}:v\in P,P\cap R=\varnothing} f_P > 0$ remaining flow pass through it) is put into R, which is in line 15. Each time that line 15 is called in layer i, the flow decreased by at most c_i. We also know that

SAMPLE(i, R) will be called at most $\prod_{j=i+1}^{l} \lceil \frac{4 \ln n}{p_j} \rceil$ times, and line 15 will be called $\lceil \frac{4 \ln n}{p_i} \rceil$ times of each call to SAMPLE(i, R). Therefore the total flow will decrease at most $t \cdot \left[\sum_{i=1}^{l} \left(c_i \cdot \prod_{j=i}^{l} \lceil \frac{4 \ln n}{p_j} \rceil \right) \right] < f$. Thus $\sum_{P \in \mathcal{P}: P \cap R = \varnothing} f_P > 0$ after we run SAMPLE(l, R) t times.

Therefore from Claim 2.3.5, we can always perform line 5 and line 8.

Line 23 fails only when $S' = \varnothing$. We know that in SAMPLE(i, R), if all the former procedure succeeded, then $|S| = \lceil \frac{4 \ln n}{p_i} \rceil$. We also know that if $p_i \geq 1$, then $S' \neq \varnothing$ with probability 1, and if $p_i < 1$, the probability that there is no good node is $(1 - p_i)^{\lceil \frac{4 \ln n}{p_i} \rceil} \leq e^{-4 \ln n} \leq \frac{1}{n^4}$. Thus $S' \neq \varnothing$ with probability at least $1 - \frac{1}{n^4}$. Because each node can be sample only once, by union bound, S' always non-empty with probability at least $1 - \frac{1}{n^3}$. Therefore with probability at least $1 - \frac{1}{n^3}$, we can perform line 23 on every call. □

After combining Claim 2.3.4, Claim 2.3.6, and the property that all the paths must be disjoint (which is mentioned in proof of Claim 2.3.4), we know that with probability at least $1 - \frac{1}{n^3}$, there are t disjoint paths from L_1 to L_l where every node in these paths is good.

2.3.2.2 Main Correctness: Proof of Theorem 2.2.3

As discussed in Section 2.2, we only need to prove that E_2 settles thin edges with high probability. So let (s, t) be a thin edge. We begin by partitioning $\mathcal{P}_{s,t}$ into four sets: for $i \in \{1, 2, 3, 4\}$, let $\mathcal{P}_{s,t}^{(i)}$ be the paths in $\mathcal{P}_{s,t}$ of length i. Note that since $\sum_{P \in \mathcal{P}_{s,t}} f_P \geq 1$, there is some i such that $\sum_{P \in \mathcal{P}_{s,t}^{(i)}} f_P \geq 1/4$.

We first show that we can find a layered graph such that there is still a

significant amount of flow on paths that obey the layering, even after the Poisson sampling.

Lemma 2.3.7. *Suppose that $\sum_{P \in \mathcal{P}_{s,t}^{(i)}} f_P \geq 1/4$. Then with probability at least $1 - e^{-\frac{25n}{36\beta}}$ we can find a layered subgraph $\hat{G}_{s,t}$ of G of the form $s \to S_1 \to \ldots \to S_{i-1} \to t$ of G with the following properties, where $\mathcal{P}'$ is the set of all possible $s-t$ paths of length i in this layered graph:*

1. *We can assign each $P \in \mathcal{P}'$ a flow value $\hat{f}_P$ such that $\sum_{P \in \mathcal{P}'} \hat{f}_P \geq \frac{1}{108}$, and*
2. *$\sum_{P \in \mathcal{P}' : e \in P} \hat{f}_P \leq \hat{x}_e$ for all $e \in E$, and*
3. *Each $\hat{f}_P$ is multiple of $\frac{\beta}{n}$.*

Proof. We begin by randomly partitioning $V_{s,t} \backslash \{s, t\}$ to $i-1$ parts $S_1, \ldots, S_{i-1}$. Then the probability that each vertex is in partition S_j is $\frac{1}{i-1}$ for $j = 1, \ldots, i-1$. Let $\mathcal{P}' \subseteq \mathcal{P}_{s,t}^{(i)}$ be the collection of paths of the form $s \to u_1 \to u_2 \to \cdots \to u_{i-1} \to t$ where $u_j \in S_j$. Clearly each path $P \in \mathcal{P}_{s,t}^{(i)}$ is in $\mathcal{P}'$ with probability $(\frac{1}{i-1})^{(i-1)} \geq \frac{1}{27}$. So $\mathbb{E}[\sum_{P \in \mathcal{P}'} f_P] \geq \frac{1}{27} \cdot \frac{1}{4} = \frac{1}{108}$, and hence there is *some* partition $S_1, \ldots, S_{i-1}$ where $\sum_{P \in \mathcal{P}'} f_P \geq \frac{1}{108}$. Fix this partition and collection of paths $\mathcal{P}'$.

Let $E_{s,t}^*$ denote the edges which are on at least one path in $\mathcal{P}'$, and let the layered graph $\hat{G}_{s,t} = (V_{s,t}, E_{s,t}^*)$. Note that every $s-t$ path in $\hat{G}_{s,t}$ has length exactly i, so the maximum flow restricted to length i paths is equal to the unrestricted maximum flow. Hence by the max-flow min-cut theorem and the fact that $\sum_{P \in \mathcal{P}'} f_P \geq \frac{1}{108}$ we know that for any $s-t$ cut $T \subseteq E_{s,t}^*$ the original capacities satisfy $\sum_{e \in T} x_e \geq \frac{1}{108}$. When we switch to the new capacities, by

Lemma 2.3.1 we get that

$$\begin{aligned}
\mathbf{Pr}\left[\sum_{e\in T}\hat{x}_e \le \frac{1}{108}\right] &\le \mathbf{Pr}\left[\sum_{e\in T} p_e\frac{\beta}{n} \le \sum_{e\in T} x_e\right] \\
&\le \mathbf{Pr}\left[\sum_{e\in T} p_e \le \sum_{e\in T}\frac{\lambda_e}{C_1}\right] \\
&\le \mathbf{Pr}\left[\sum_{e\in T} p_e \le \frac{1}{e}\sum_{e\in T}\lambda_e\right] \\
&\le e^{-\frac{1}{4}\sum_{e\in T}\lambda_e} \le e^{-\frac{C_1 n}{4\beta}\sum_{e\in T} x_e} \le e^{-\frac{25n}{18\beta}}
\end{aligned}$$

Since (s,t) is a thin edge we know that $|V_{s,t}| \le \frac{n}{\beta}$, and hence there are at most $2^{\frac{n}{\beta}}$ cuts. A union bound then implies that the probability that there is *any* cut T with $\sum_{e\in T}\hat{x}_e \le \frac{1}{108}$ is at most $e^{-\frac{25n}{18\beta}+\frac{n}{\beta}\ln 2} \le e^{-\frac{25n}{36\beta}}$. Therefore with probability at least $1-e^{-\frac{25n}{36\beta}}$, the maximum flow from s to t in $\hat{G}_{s,t}$ with capacities $\{\hat{x}_e\}$ is at least $\frac{1}{108}$ (by max-flow min-cut). Also, because the new capacity of each edge is multiple of $\frac{\beta}{n}$, we can let the flow of each path be multiple of $\frac{\beta}{n}$, which implies the lemma. □

We now have all of the lemmas necessary to prove that thin edges with significant flow along paths of length 4 are settled by E_2.

Lemma 2.3.8. *Let (s,t) be a thin edge with $\sum_{P\in\mathcal{P}^{(4)}_{s,t}} f_P \ge \frac{1}{4}$. Then E_2 settles (s,t) with probability at least $1-\frac{2}{n^3}$.*

Proof. We first use Lemma 2.3.7 to get the layered graph $\hat{G}_{s,t} = (V_{s,t}, E^*_{s,t})$ and the path set $\mathcal{P}'$ in $\hat{G}_{s,t}$ (which by construction each have length exactly 4). From Lemma 2.3.7 we know that the total flow $\sum_{P\in\mathcal{P}'}\hat{f}_P$ is at least $\frac{1}{108}$ (with

very high probability).

For each $e \in E^*_{s,t}$, let $\hat{f}_e = \sum_{P \in \mathcal{P}': e \in P}$ denote the amount of flow which uses edge e (so clearly $\hat{f}_e \leq \hat{x}_e$), and for each $V \in V_{s,t}$, define the *load* l_v to be $l_v = \sum_{P \in \mathcal{P}': v \in P} \hat{f}_P = \sum_{u:(u,v) \in E^*_{s,t}} \hat{f}_{(u,v)}$, which is the total flow which passes through v.

Note that $\hat{G}_{s,t}$ is a directed acyclic graph, which was partitioned to layers s, S_1, S_2, S_3, t. We just need to show that E_2 contains a path through $s \to S_1 \to S_2 \to S_3 \to t$ with high probability.

We first split the problem into two cases, depending on whether there is a node in S_2 with large load.

Case 1: Suppose that there is a node v in S_2 with $l_v \geq \frac{1}{\beta}$. Consider an arbitrary path $v \to w \to t$, with $w \in S_3$. Since all flow goes from S_3 to t, we know that $\hat{f}_{(w,t)} = l_w \geq \hat{f}_{(v,w)}$. Thus a sufficient condition for there to be a path from v to t in E_2 is for there to be some $w \in S_3$ with $r_w \leq \alpha \hat{f}_{(v,w)}$, since that would guarantee the inclusion of (v, w) and (w, t). Hence the probability of there *not* existing a path from v to t in E_2 is at most

$$\begin{aligned}
\prod_{w \in S_3} \min(1 - \alpha \hat{f}_{(v,w)}, 0) &\leq \prod_{w \in S_3} e^{-\alpha \hat{f}_{(v,w)}} \\
&= e^{-\alpha \cdot \sum\limits_{w \in S_3} \hat{f}_{(v,w)}} \\
&= e^{-C_2 \beta \ln^6 n \cdot l_v} \leq e^{-\ln^6 n}
\end{aligned}$$

By the same argument, the probability that there does not exist a path (of length 2) from s to v is also at most $e^{-\ln^6 n}$. Therefore with probability at least

$1 - 2e^{-\ln^6 n}$, there is a path from s to t of length 4 in E_2.

Case 2: Suppose that there is no node in S_2 which has load at least $\frac{1}{\beta}$. For each path $P \in \mathcal{P}'$, let u_P be the node of P in S_1, let v_P be the node of P in S_2, and let w_P be the node of P in S_3 (so P is of the form $s \to u_P \to v_P \to w_p \to t$). We will bucket the paths in $\mathcal{P}'$ according to four values: l_{u_P}, $\hat{f}_{(u_P,v_P)}$, $\hat{f}_{(v_P,w_P)}$, and l_{w_P}. Note that all four of these values are multiples of β/n (due to the Poisson sampling), and are at most 1. Let $B_{i,j,k,\ell}$ be the bucket which contains all paths $P \in \mathcal{P}$ where $2^i\frac{\beta}{n} \leq l_{u_P} < 2^{i+1}\frac{\beta}{n}$ and $2^j\frac{\beta}{n} \leq \hat{f}_{(u_P,v_P)} < 2^{j+1}\frac{\beta}{n}$ and $2^k\frac{\beta}{n} \leq \hat{f}_{(v_P,w_P)} < 2^{k+1}\frac{\beta}{n}$ and $2^\ell\frac{\beta}{n} \leq l_{w_P} < 2^{\ell+1}\frac{\beta}{n}$. Note that there are only $(\log\frac{n}{\beta})^4$ buckets, and hence there must be a bucket $B_{i,j,k,\ell}$ such that $\sum_{P\in B_{i,j,k,\ell}} \hat{f}_P \geq \frac{\frac{1}{108}}{(\log\frac{n}{\beta})^4} \geq \frac{1}{500\ln^4 n}$ (since $\beta = n^{\frac{1}{3}}$).

In order to simplify notation, we will fix this bucket and let $B = B_{i,j,k,\ell}$, let $a = 2^i\frac{\beta}{n}$, let $b = 2^\ell\frac{\beta}{n}$, let $c = 2^j\frac{\beta}{n}$, and let $d = 2^k\frac{\beta}{n}$. So for each $P \in B$ we have that $a \leq l_{u_P} < 2a$, $b \leq l_{w_P} < 2b$, $c \leq \hat{f}_{(u_P,v_P)} < 2c$, and $d \leq \hat{f}_{(v_P,w_P)} < 2d$.

We further split the problem to 4 cases, depending on whether $a \geq \frac{1}{\alpha}$ and whether $b \geq \frac{1}{\alpha}$:

Case 2.1: $a < \frac{1}{\alpha}$ and $b < \frac{1}{\alpha}$.

Without loss of generality, we can assume $\frac{c}{a} \leq \frac{d}{b}$ (since otherwise we can simply consider the graph from the reverse direction). Let $P \in B$, and note that if

$$r_{u_P} \leq a\alpha \text{ and } r_{w_P} \leq b\alpha \text{ and } r_{v_P} \leq \frac{c}{a} \tag{2.2}$$

then all edges of P are in E_2, and so E_2 settles (s,t). This is easy to see by simple calculations: if (2.2) holds, then $r_s r_{u_P} \leq r_{u_P} \leq a\alpha \leq l_{u_P}\alpha \leq \hat{x}_{(s,u_P)}\alpha$ and so $(s,u_P) \in E_2$. Similarly, at the other end of the path we would have

that $r_{w_P} r_t \leq r_{w_P} \leq b\alpha \leq l_{w_P}\alpha \leq \hat{x}_{(w_P,t)}\alpha$, so $(w_P, t) \in E_2$. For the second edge of the path, we would have that $r_{u_P} r_{v_P} \leq a\alpha \cdot \frac{c}{a} = c\alpha \leq \hat{x}_{(u_P,v_P)}\alpha$, so $(u_P, v_P) \in E_2$. Finally, for the third path edge we would have that $r_{v_P} r_{w_P} \leq \frac{c}{a} \cdot b\alpha \leq \frac{d}{b} \cdot b\alpha = d\alpha \leq \hat{x}_{(v_P,w_P)}\alpha$, and so $(v_P, w_P) \in E_2$.

We use Lemma 2.2.4, treating S_3 as layer 1, S_2 as layer 2, S_1 as layer 3 in the lemma. We know that $c_1 = 2b$, $c_2 = \frac{1}{\beta}$, $c_3 = 2a$. Let $p_1 = b\alpha$, $p_2 = \frac{c}{a}$, $p_3 = a\alpha$. Then, in order to prove that there exists a good path $P \in E_2$ with high probability, we only need to prove $\sum_{i=1}^{3} \left(c_i \cdot \prod_{j=i}^{3} \lceil \frac{4 \ln n}{p_j} \rceil \right) < \frac{1}{500 \ln^4 n}$. And because $c \geq \frac{\beta}{n}, d \geq \frac{\beta}{n}, p_1 < 1, p_2 < 1, p_3 < 1$, we get that

$$
\begin{aligned}
&\sum_{i=1}^{3} \left(c_i \cdot \prod_{j=i}^{3} \left\lceil \frac{4 \ln n}{p_j} \right\rceil \right) \\
&< 2b \cdot \frac{5 \ln n}{b\alpha} \cdot \frac{5 \ln n}{\frac{c}{a}} \cdot \frac{5 \ln n}{a\alpha} + \frac{1}{\beta} \cdot \frac{5 \ln n}{\frac{c}{a}} \cdot \frac{5 \ln n}{a\alpha} + 2a \cdot \frac{5 \ln n}{a\alpha} \\
&= \frac{250 \ln^3 n}{c\alpha^2} + \frac{25 \ln^2 n}{d\alpha\beta} + \frac{10 \ln n}{\alpha} \\
&\leq \frac{250 \ln^3 n}{\frac{\beta}{n}(C_2 \beta \ln^6 n)^2} + \frac{25 \ln^2 n}{\frac{\beta}{n}(C_2 \beta \ln^6 n)\beta} + \frac{10 \ln n}{(C_2 \beta \ln^6 n)} \\
&< \frac{1}{500 \ln^4 n}
\end{aligned}
$$

Therefore, with probability at least $1 - \frac{1}{n^3}$, there is a path $P \in E_2$ from s to t with length 4.

Case 2.2: $a \geq \frac{1}{\alpha}$ and $b < \frac{1}{\alpha}$.

As in case 2.1, let $P \in B$, and note that if

$$r_{w_P} \leq b\alpha \text{ and } r_{v_P} \leq \frac{\alpha\beta}{n} \tag{2.3}$$

then all edges of P are in E_2, and so E_2 settles (s,t). This is because if (2.3) holds, then $r_s r_{u_P} \leq 1 \leq a\alpha \leq l_{u_P}\alpha \leq \hat{x}_{(s,u_P)}\alpha$ and so $(s,u_P) \in E_2$. Similarly, at the other end of the path we would have that $r_{w_P} r_t \leq r_{w_P} \leq b\alpha \leq l_{w_P}\alpha \leq \hat{x}_{(w_P,t)}\alpha$, so $(w_P,t) \in E_2$. For the second edge of the path, we would have that $r_{u_P} r_{v_P} \leq r_{v_P} \leq \frac{\alpha\beta}{n} \leq c\alpha \leq \hat{x}_{(u_P,v_P)}\alpha$, so $(u_P,v_P) \in E_2$. Finally, for the third path edge we would have that $r_{v_P} r_{w_P} \leq r_{v_P} \leq \frac{\alpha\beta}{n} \leq d\alpha \leq \hat{x}_{(v_P,w_P)}\alpha$, and so $(v_P,w_P) \in E_2$.

We again use Lemma 2.2.4, but this time only need to use it for two levels. We treat S_3 as layer 1 and S_2 as layer 2 in the lemma, and we know that $c_1 = 2b$, $c_2 = \frac{1}{\beta}$. Let $p_1 = b\alpha$, $p_2 = \frac{\alpha}{n}$. Then, in order to prove that there exists a good path $P \in E_2$ with high probability, we only need to prove $\sum_{i=1}^{2}\left(c_i \cdot \prod_{j=i}^{2}\lceil\frac{4\ln n}{p_j}\rceil\right) < \frac{1}{500\ln^4 n}$. And because $p_1 < 1, p_2 < 1, p_3 < 1$, we get that

$$\begin{aligned}
\sum_{i=1}^{2}\left(c_i \cdot \prod_{j=i}^{2}\left\lceil\frac{4\ln n}{p_j}\right\rceil\right) &< 2b \cdot \frac{5\ln n}{b\alpha} \cdot \frac{5\ln n}{\frac{\alpha\beta}{n}} + \frac{1}{\beta} \cdot \frac{5\ln n}{\frac{\alpha\beta}{n}} \\
&= \frac{50n\ln^2 n}{\alpha^2\beta} + \frac{5n\ln n}{\alpha\beta^2} \\
&\leq \frac{50n\ln^2 n}{(C_2\beta\ln^6 n)^2\beta} + \frac{5n\ln n}{(C_2\beta\ln^6 n)\beta^2} < \frac{1}{500\ln^4 n}
\end{aligned}$$

Therefore, with probability at least $1 - \frac{1}{n^3}$, there is a path $P \in E_2$ from s to

t with length 4.

Case 2.3: $a < \frac{1}{\alpha}$ and $b \geq \frac{1}{\alpha}$.

This case is symmetric to case 2.2. Let $P \in B$, and note that if

$$r_{u_P} \leq a\alpha \text{ and } r_{v_P} \leq \frac{\alpha\beta}{n} \tag{2.4}$$

then all edges of P are in E_2, and so E_2 settles (s,t).

We use Lemma 2.2.4, treating S_1 as layer 1 and S_2 as layer 2. Then essentially the same calculation as in case 2.2 implies that with probability at least $1 - \frac{1}{n^3}$, there is a path $P \in E_2$ from s to t with length 4.

Case 2.4: $a \geq \frac{1}{\alpha}$ and $b \geq \frac{1}{\alpha}$.

This is actually the simplest case. Let $P \in B$, and note that if

$$r_{v_P} \leq \frac{\alpha\beta}{n} \tag{2.5}$$

then all edges of P are in E_2, and so E_2 settles (s,t). This is because if (2.5) holds, then $r_s r_{u_P} \leq 1 \leq a\alpha \leq l_{u_P}\alpha \leq \hat{x}_{(s,u_P)}\alpha$ and so $(s,u_P) \in E_2$. Similarly, at the other end of the path we would have that $r_{w_P} r_t \leq 1 \leq b\alpha \leq l_{w_P}\alpha \leq \hat{x}_{(w_P,t)}\alpha$, so $(w_P,t) \in E_2$. For the second edge of the path, we would have that $r_{u_P} r_{v_P} \leq r_{v_P} \leq \frac{\alpha\beta}{n} \leq c\alpha \leq \hat{x}_{(u_P,v_P)}\alpha$, so $(u_P,v_P) \in E_2$. Finally, for the third path edge we would have that $r_{v_P} r_{w_P} \leq r_{v_P} \leq \frac{\alpha\beta}{n} \leq d\alpha \leq \hat{x}_{(v_P,w_P)}\alpha$, and so $(v_P,w_P) \in E_2$.

Because there is no node in S_2 which has load at least $\frac{1}{\beta}$ and the flow through B is at least $\frac{1}{500\ln^4 n}$, there are at least $\frac{\beta}{500\ln^4 n}$ nodes in S_2 which are in a path of B. Thus there is a node v_P such that $r_{v_P} \leq \frac{\alpha\beta}{n}$ with probability at least $1 - (1 - \frac{\alpha\beta}{n})^{\frac{\beta}{500\ln^4 n}} \geq 1 - e^{\frac{\alpha\beta^2}{500n\ln^4 n}} \geq 1 - e^{-\ln^2 n}$.

Therefore, with probability at least $1 - e^{-\ln^2 n}$, there is a path $P \in E_2$ from s to t with length 4.

By a union bound on the probability that total flow is at least $\frac{1}{108}$ and the probability in each case before, with probability at least $1 - \frac{1}{n^3} - e^{\frac{25n}{36\beta}} \geq 1 - \frac{2}{n^3}$, E_2 contains a length 4 path for thin edge (s,t) if $\sum_{P \in \mathcal{P}_{s,t}^{(4)}} f_P^* \geq \frac{1}{4}$. Therefore we have proved the lemma. □

Finally, we can prove our main correctness theorem, which in turn implies Theorem 2.2.3.

Theorem 2.3.9. *E_2 settles all thin edges with probability at least $1 - \frac{2}{n}$.*

Proof. Let (s,t) be a thin edge. Since $\sum_{P \in \mathcal{P}_{s,t}} f_P \geq 1$, there is some $i \in \{1,2,3,4\}$ such that $\sum_{P \in \mathcal{P}_{s,t}^{(i)}} \geq 1/4$. If this is true for some $i \in \{1,2,3\}$, then we know from (Berman et al., 2013) that E_2 settles (s,t) with probability at least $1 - \frac{1}{n^3}$ (since any edge picked by their algorithm is also picked in E_2). Otherwise we know from Lemma 2.3.8 that E_2 settles (s,t) with probability at least $1 - \frac{2}{n^3}$. A union bound completes the proof. □

2.4 Integrality Gap

The basic flow LP (2.1) (or simple variations of it) has been used in the best-known approximations of many spanner problems, including DIRECTED k-SPANNER and BASIC 3-SPANNER (Berman et al., 2013), LOWEST DEGREE k-SPANNER (Chlamtác and Dinitz, 2014; Chlamtác, Dinitz, and Krauthgamer,

2012), and various versions of f-FAULT-TOLERANT k-SPANNER (Dinitz and Krauthgamer, 2011a; Dinitz and Krauthgamer, 2011b).

Given the prevalence and usefulness of this LP, it is natural to attempt to use it to break the trivial upper bound from (Althöfer et al., 1993) for BASIC k-SPANNER. In this section we show an integrality gap which proves that only very limited improvements are possible when using this LP.

We begin by proving the existence of instances of MIN-REP with certain parameters (large OPT, large supergirth, bijection constraints). These instances will then be used to construct our integrality gap.

As with our algorithm and analysis for 4-spanner, we first give an overview in this section of the main techniques, and then in Section 2.5 give the complete construction and analysis.

2.4.1 MIN-REP Instances

MIN-REP was originally defined by Kortsarz (Kortsarz, 2001) as a tool for proving hardness of spanner problems. It is essentially a minimization version of the well-known LABEL COVER problem. In MIN-REP we are given a bipartite graph $G = (A, B, E)$ where A is partitioned into groups $A_1, A_2, \ldots, A_r$ and B is partitioned into groups $B_1, B_2, \ldots, B_r$, with the additional property that every set A_i and every set B_j has the same size (which we will call $|\Sigma|$ due to its connection to the alphabet of a 1-round 2-prover proof system). This graph and partition induces another bipartite graph $G' = (U, V, E')$ called the *supergraph* in which there is a vertex $a_i \in U$ for each group A_i and similarly a vertex $b_j \in V$ for each group B_j. There is an edge between a_i and b_j in G' if

there is an edge in G between some node in A_i and some node in B_j. A node in G' is called a supernode, and similarly an edge in G' is called a superedge.[2] A REP-cover is a set $C \subseteq A \cup B$ with the property that for all superedges $\{a_i, b_j\}$ there are nodes $a \in A_i \cap C$ and $b \in B_j \cap C$ where $\{a, b\} \in E$. We say that $\{a, b\}$ *covers* the superedge $\{a_i, b_j\}$. The goal is to construct a REP-cover of minimum size.

Instead of using MIN-REP as the starting point of a hardness reduction (in which case there is a very tight connection with PCPs and 1-round 2-prover proof systems), we will use it as the starting point of an integrality gap. Hence we allow ourselves to build instances with structure that will help the gap, rather than being given by a proof system. These instances will not necessarily be computationally hard, but instead will have large integrality gaps and useful structural properties.

Most importantly, we will need instances with large *supergirth* (the size of the smallest cycle in the supergraph). So to begin with, we will assume that the Erdős girth conjecture is true for bipartite graphs. In particular, we will assume the following:

Conjecture 2.4.1. *For any integer k, for large enough n there are bipartite graphs with n nodes on each side and girth at least $2k + 2$ that are regular with degree $d = \Omega(n^{1/k})$.*

This conjecture has been made by Erdős (Erdős, 1964) and by Bondy and

[2]Rather than G being the graph and G' being the supergraph, sometimes G' is referred to as the graph and G is called the *label-extended graph*.

Simonovits (Bondy and Simonovits, 1974)[3]. If Conjecture 2.4.1 is false, then we can instead fall back on the best known constructions of dense, large girth graphs (Lazebnik, Ustimenko, and Woldar, 1995). These do not quite achieve degree $d = \Omega(n^{1/k})$ for girth at least $2k+2$, but come somewhat close: they have degree approximately $n^{2/(3k)}$ (this a simplification of a more complex bound – see (Lazebnik, Ustimenko, and Woldar, 1995) or Thorup and Zwick, 2005, Table 2 for the fully correct bound). In general, if there are graphs with girth at least $2k+2$ and degree n^α for some $\alpha \leq 1/k$, then wherever k appears in the rest of this section it can be replaced by $1/\alpha$.

This will now let us build instances of MIN-REP that have both large supergirth and large OPT.

Theorem 2.4.2. *For any integer k, assuming Conjecture 2.4.1 there is an infinite family of instances of* MIN-REP *with the following properties:*

1. $|U| = |V| = n$,
2. *Every supervertex has degree $d = \Omega(n^{1/k})$ in the supergraph G',*
3. *G' has girth at least $2k+2$,*
4. $|\Sigma| = n^\gamma$ *for* $\gamma = \frac{2}{k} + \delta$ *where* $\delta > 0$ *is an arbitrarily small constant,*
5. *If $\{a_i, b_j\} \in E'$ then the edges between A_i and B_j in G form a perfect matching, and*

[3]The girth conjecture is sometimes stated without the regularity and bipartiteness conditions. The bipartiteness restriction is without loss of generality thanks to standard graph transformations. The regularity condition is not without loss of generality, but the best known constructions of dense, large girth graphs (Lazebnik, Ustimenko, and Woldar, 1995) are regular. Moreover, it is easy to see that we can assume near-regularity (degrees all within a logarithmic factor) without loss of generality (other than a logarithmic factor in the total number of edges), and that this suffices for the claimed integrality gap.

6. $OPT \geq \Omega(dn) = \Omega(n^{1+1/k})$.

The intuition behind the proof of Theorem 2.4.2 is to use the probabilistic method. We start with the graphs from Conjecture 2.4.1 as the supergraph G', and for each superedge we construct a random matching in G. To prove that OPT is large, we consider each set of size less than dn and prove that the probability that it is a REP-cover is extremely small. We then do a union bound over all of these small sets to get that the probability that *any* small set is a REP-cover is less than 1. Now the probabilistic method implies that there is *some* instance where OPT is large.

2.4.2 BASIC k-SPANNER

Now that we have instances of MIN-REP with large supergirth and large OPT, we can construct the integrality gap instances for BASIC k-SPANNER. They essentially follow from applying the hardness reduction of (Kortsarz, 2001; Elkin and Peleg, 2000) to the instances from Theorem 2.4.2. We include the details for completeness in Section 2.5.

Let OPT_{MR} denote the optimal value of the MIN-REP instances from Theorem 2.4.2. At a high level, the reduction of (Kortsarz, 2001; Elkin and Peleg, 2000) constructs a graph H with the property that if OPT_{MR} is large then any k-spanner of H must be large. In order for this to be true we need the MIN-REP instance to have supergirth larger than $k+1$, which is why we required the supergirth property in Theorem 2.4.2 (MIN-REP was shown to be hard even with large supergirth by (Dinitz, Kortsarz, and Raz, 2012), but we need extremely large OPT_{MR}, not hardness).

The LP, on the other hand, can find a fractional solution of cost only $O(|V(H)|)$. While the LP is for BASIC k-SPANNER on H, this fractional solution intuitively corresponds to a simple fractional solution to the MIN-REP instance it is based on, which just includes *every* node in the REP-cover but only fractionally at $1/|\Sigma|$. Since each superedge $\{a_i, b_j\}$ is a matching in G between A_i and B_j, including all nodes at $1/|\Sigma|$ satisfies the covering requirement (fractionally, it looks the same as picking one node in A_i and one node in B_j that are adjacent).

By setting parameters carefully, we can get both of these properties (large OPT, small fractional solution) to hold simultaneously, proving Theorem 2.1.2.

2.5 Details of Integrality Gap

We begin by formally proving the existence of MIN-REP instances with the properties we need, and then show how to use these instances to build the integrality gap.

2.5.1 MIN-REP Instances: Proof of Theorem 2.4.2

We start with the graphs from Conjecture 2.4.1. We will construct random instances based on these graphs, and show by the probabilistic method that there is some instance where OPT is large. Let $G' = (U, V, E')$ be one of these graphs with $|U| = |V| = n$, and note that parts 1, 2, and 3 of the theorem are true simply because we are using the graphs from Conjecture 2.4.1, so it simply remains to prove parts 4, 5, and 6.

For each edge $e = \{a_i, b_j\} \in E'$, let $\pi_e : A_i \to B_j$ be a bijection chosen

uniformly at random from the space of all bijections. Note that now by construction we have satisfied part 5 of the theorem. We claim that with nonzero probability, $OPT \geq \epsilon dn$. To see this, consider a set $L \subseteq A \cup B$ with $|L| = 2\alpha n$, making the average number of nodes per supernode α (smaller values of $|L|$ are even easier to handle, so for simplicity we ignore them). We will call nodes in L *selected* nodes. We will upper bound the probability that L is a REP-cover when we set $\alpha = \epsilon d$.

Since the average number of nodes in L per supernode is α and there are $2n$ supernodes, by Markov's inequality (or a simple averaging argument) we know that at most $n/2$ supernodes have more than 4α nodes in L. Call these supernodes *heavy*, and call all other supernodes *light*. Thus at most $d(n/2)$ superedges are incident on heavy supernodes, so at least $d(n/2)$ superedges have two light endpoints. We say that such a superedge is light.

Let $\{a_i, b_j\}$ be a light superedge, and let $\alpha \in A_i \cap L$ be an arbitrary selected node in A_i. Since the bijection $\pi_{\{a_i,b_j\}}$ is chosen uniformly at random, the probability that $\pi_{\{a_i,b_j\}}(\alpha) \in L \cap B_j$ is at most $|L \cap B_j|/|\Sigma| \leq 4\alpha/|\Sigma|$. Since $|L \cap A_i| \leq 4\alpha$, a union bound implies that $\mathbf{Pr}[L \text{ covers } \{a_i, b_j\}] \leq 16\alpha^2/|\Sigma|$. Since in order for L to be a REP-cover it must cover all of the light superedges and the bijection for each superedge is chosen independently, we get that

$$\mathbf{Pr}[L \text{ a REP-cover}] \leq \left(\frac{16\alpha^2}{|\Sigma|}\right)^{d\frac{n}{2}} = \frac{(4\alpha)^{dn}}{|\Sigma|^{dn/2}}. \tag{2.6}$$

We now bound the number of possible sets L, which is straightforward: out of $2n \cdot |\Sigma|$ total nodes, we required L to be a set of $2\alpha n$. Thus the number

of possible sets is at most

$$\binom{2n|\Sigma|}{2\alpha n} \leq \left(\frac{2n|\Sigma|e}{2\alpha n}\right)^{2\alpha n} = \left(\frac{e|\Sigma|}{\alpha}\right)^{2\alpha n}. \tag{2.7}$$

Now a union bound, equations (2.6) and (2.7), and setting $\alpha = \epsilon d$ together imply that

$$\begin{aligned}
\mathbf{Pr}\left[OPT \leq 2\alpha n\right] &\leq \left(\frac{e|\Sigma|}{\alpha}\right)^{2\alpha n} \cdot \frac{(4\alpha)^{dn}}{|\Sigma|^{dn/2}} \\
&= e^{2\alpha n} \cdot 4^{dn} \cdot \frac{\alpha^{dn-2\alpha n}}{|\Sigma|^{(dn/2)-2\alpha n}} \\
&= e^{2\epsilon dn} 4^{dn} \cdot \frac{(\epsilon d)^{(1-2\epsilon)dn}}{|\Sigma|^{(\frac{1}{2}-2\epsilon)dn}} \\
&\leq (4e)^{dn} \cdot \frac{d^{(1-2\epsilon)dn}}{|\Sigma|^{(\frac{1}{2}-2\epsilon)dn}} \\
&= (4e)^{n^{1+1/k}} \cdot \frac{n^{\frac{1}{k}(1-2\epsilon)n^{1+1/k}}}{|\Sigma|^{(\frac{1}{2}-2\epsilon)n^{1+1/k}}}.
\end{aligned}$$

If this is less than 1, then we know that there must exist some instantiation of the randomness such that $OPT > 2\alpha n = 2\epsilon dn = 2\epsilon n^{1+1/k}$, which would imply the final part of the theorem. Thus we need $|\Sigma| = n^{\gamma}$ to be large enough to make the ratio less than 1. By making ϵ a very small constant and n large enough, we get that it is sufficient for γ to be $\frac{2}{k} + \delta$, where δ is an arbitrarily small constant that depends on ϵ. This finishes the proof of the theorem.

2.5.2 BASIC k-SPANNER Instance

We now show how to use Theorem 2.4.2 to build integrality gap instances for BASIC k-SPANNER. For ease of notation, we will actually consider BASIC $(2k-1)$-SPANNER, and only convert the stretch at the end.

Let $G = (A, B, E)$ and $G' = (U, V, E')$ be the graph and supergraph for an instance from Theorem 2.4.2. Let $x = n^{\gamma} = n^{\frac{2}{k}+\delta}$. We will create a new graph $H = (V_H, E_H)$ which will be the integrality gap instance. To define V_H, we first need two new sets of vertices:

$$S = \{s_{ij}^{p} : a_i \in U, j \in [k-1], p \in [x]\}$$

$$T = \{t_{ij}^{p} : b_i \in V, j \in [k-1], p \in [x]\}$$

The vertex set of H will be $V_H = A \cup B \cup S \cup T$. Note that $|V_H| = 2n^{1+\gamma} + 2(k-1)xn = 2kn^{1+\gamma} = 2kn^{1+\frac{2}{k}+\delta}$.

We now define a collection of edge sets:

$$E_M = \{\{s^p_{ij}, s^p_{i(j+1)}\} : p \in [x], a_i \in U, j \in [k-2]\}$$

$$\cup \{\{t^p_{ij}, t^p_{i(j+1)}\} : p \in [x], b_i \in V, j \in [k-2]\},$$

$$E_{sA} = \{\{s^p_{i1}, a\} : a_i \in U, a \in A_i, p \in [x]\},$$

$$E_{tB} = \{\{b, t^p_{j1}\} : b_j \in V, b \in B_j, p \in [x]\},$$

$$E^p_{G'} = \{\{s^p_{i(k-1)}, t^p_{j(k-1)}\} : a_i \in U, b_j \in V, (a_i, b_j) \in E'\},$$

$$E_{G'} = \cup^x_{p=1} E^p_{G'},$$

$$E_C = (\cup^n_{i=1}\{\{a, a'\} : a, a' \in A_i\}) \bigcup (\cup^n_{i=1}\{\{b, b'\} : b, b' \in B_i\}).$$

Our final edge set E_H is the union of all of these edges together with E. Intuitively, H consists of a single copy of G and x copies of G' (for each $p \in [x]$ the edges $E^p_{G'}$ form a graph isomorphic to G'). For each supernode in G' and each $p \in [x]$ there is one vertex incident to the edges in G', then a path of length $k-2$, and then a vertex adjacent to the nodes in the group of the supernode.

For two nodes $s^p_{i(k-1)}$ and $t^p_{j(k-1)}$ with $\{a_i, b_j\} \in E'$, it is easy to verify that (other than the direct edge between them) there are exactly n^γ paths of length at most $2k-1$ between them. These paths are called the *canonical paths* for the pair, and all have the form

$$s^p_{i(k-1)} \to s^p_{i(k-2)} \to \cdots \to s^p_{i1} \to a \to b \to t^p_{j1} \to t^p_{j2} \to \cdots \to t^p_{j(k-1)}$$

where $a \in A_i$ and $b \in B_j$ and $\{a, b\} \in E$. Note that there are exactly n^γ such paths (corresponding to the choice of edge in E) and they all have different nodes in A_i, B_j (due to the perfect matching between A_i and B_j).

In fact, this is exactly why we need instances of MIN-REP which have large girth – if the girth is less than $2k+1$, then there is the possibility of $\{s^p_{i(k-1)'}, t^p_{j(k-1)}\}$ being spanned by a path consisting of edges from $E_{G'}$. But without such paths, the only way to span the edge is to use a canonical path or include the edge itself.

We first claim that there is a small fractional solution.

Theorem 2.5.1. *There is a fractional solution to the flow LP on H with objective value $O(|V_H|)$.*

Proof. Set $x_e = 1$ for all edges $e \in E_M$, and set $x_e = 0$ for all $e \in E_{G'}$. Set $x_e = 2/n^\gamma$ for all other edges in E_H. Under this assignment, the objective function of the LP is

$$\begin{aligned}
\sum_{e \in E_H} x_e &= |E_M| + \frac{2}{n^\gamma}\left(|E| + |E_{sA}| + |E_{tB}| + |E_C|\right) \\
&\leq 2xn(k-2) + \frac{2}{n^\gamma}\left(dnn^\gamma + 2xnn^\gamma + n(n^{2\gamma})\right) \\
&\leq |V_H| + 2n^{1+\frac{1}{k}} + 4n^{1+\gamma} + 2n^{1+\gamma} \\
&\leq |V_H| + O(n^{1+\gamma}) \leq O(|V_H|)
\end{aligned}$$

Hence it remains only to prove that with these values for the x variables, there are values for the y variables which form a feasible solution to the LP.

In other words, we need to show how to ship one unit of flow along paths of length $(2k-1)$ for every edge in E_H. This is trivial for edges in E_M, since they have capacity one. For edges in E_C, we can send $2/n^\gamma$ flow directly and $(2/n^\gamma)(n^\gamma - 2)$ flow along paths of length 2 through other nodes in the same group (using other edges of E_C), so we can send at least 1 unit of flow.

For each edge $(a,b) \in E$ (with $a \in A_i$ and $b \in B_j$), we can again send $2/n^\gamma$ flow directly and can send $(n^\gamma - 1)(2/n^\gamma) > 1$ flow on paths of length 3 of the form $a \to a' \to b' \to b$, where $a' \in A_i$ and $b' \in B_j$ and $\{a', b'\} \in E$. For each edge $\{s_{i1}^p, a\} \in E_{sA}$, we can again send $2/n^\gamma$ flow directly and can send another $(2/n^\gamma)(n^\gamma - 1) \geq 1$ flow on paths of length 2 of the form $s_{i-1}^p \to a' \to a$, where $a, a' \in A_i$. A similar argument holds for edges in E_{tB}.

Finally for edges in $E_{G'}$ we can simply send $1/n^\gamma$ flow along each of the n^γ canonical paths, for a total flow of 1. □

We now need to claim that the integral OPT is large.

Theorem 2.5.2. *Every $(2k-1)$-spanner of H has at least $\Omega(n^{1+\frac{3}{k}+\delta})$ edges.*

Proof. Let H' be a $(2k-1)$-spanner of H. If H' uses any edge from $E_{G'}$, we can replace this edge with an arbitrary canonical path to get a $(2k-1)$-spanner H'' of H with size at most $(2k-1)|H'|$. Now since H'' does not use any edges from $E_{G'}$, every edge in $E_{G'}$ must be spanned by at least one canonical path (since as noted they are the only paths of length $2k-1$ between the endpoints other than the edge itself).

For each $p \in [x]$, let

$$L_p = \left(\cup_{i\in[n]}\{a \in A_i : \{s_{i1}^p, a\} \in H''\}\right) \quad \bigcup \left(\cup_{i\in[n]}\{b \in B_i : \{b, t_{i1}^p\} \in H''\}\right)$$

denote the nodes from G that in H'' are adjacent to nodes in S or T for the given p. We claim that L_p is a valid REP-cover. To see this, consider an arbitrary superedge $\{a_i, b_j\}$. Then in H'' the nodes $s^p_{i(k-1)}$ and $t^p_{j(k-1)}$ are spanned by a canonical path, and hence L_p contains $a \in A_i, b \in B_j$ such that $\{a, b\} \in E$. Thus L_p is a REP-cover.

Thus $|L_p| \geq \Omega(n^{1+\frac{1}{k}})$ by Theorem 2.4.2. Hence $|H''| \geq \Omega(xn^{1+\frac{1}{k}}) = \Omega(n^{1+\frac{3}{k}+\delta})$. □

Combining Theorems 2.5.1 and 2.5.2 gives us the desired theorem.

Theorem 2.5.3. *For any constant $\delta > 0$, the integrality gap of the flow LP for* BASIC $(2k-1)$-SPANNER *is at least* $\Omega(n^{\frac{1}{(1+\delta)k+2}})$.

Proof. The only thing remaining is to rewrite the gap in terms of $|V_H|$, since we want the n of the theorem statement to correspond to the number of nodes in the integrality gap graph H rather than the number of nodes in G'. We know that $|V_H| = 2kn^{1+\frac{2}{k}+\delta}$, and hence $n \geq \Omega(|V_H|^{\frac{1}{1+(2/k)+\delta}})$. From Theorem 2.5.2 we get that every spanner of H has at least $n^{1+\frac{3}{k}+\delta} = \Omega(|V_H|^{\frac{1+(3/k)+\delta}{1+(2/k)+\delta}}) = \Omega(|V_H|^{\frac{k+3+\delta k}{k+2+\delta k}})$ edges. Now Theorem 2.5.1 implies that the integrality gap is at least $\Omega(|V_H|^{\frac{k+3+\delta k}{k+2+\delta k}-1}) = \Omega(|V_H|^{\frac{1}{(1+\delta)k+2}})$. □

2.5.3 Proof of Theorem 2.1.2

Suppose that $k = 2t-1$ is odd. Then by Theorem 2.5.3, the integrality gap is at least $\Omega(n^{\frac{1}{(1+\delta)t+2}}) = \Omega(n^{\frac{1}{(1+\delta)\frac{k+1}{2}+2}}) = \Omega(n^{\frac{2}{(1+\delta)(k+1)+4}})$ as claimed. If $k = 2t$ is even, then it is easy to verify that if we use the construction of H for $2t-1$ then there is still no way of spanning an edge in $E_{G'}$ other than the edge

itself or a canonical path (which now have length $k-1$ rather than k). Hence the integrality gap is at least $\Omega(n^{\frac{1}{(1+\delta)t+2}}) = \Omega(n^{\frac{1}{(1+\delta)\frac{k}{2}+2}}) = \Omega(n^{\frac{2}{(1+\delta)k+4}})$ as claimed.

2.6 Fault-Tolerance

As with the first two results, we break our discussion of f-FAULT-TOLERANT k-SPANNER into two sections. We begin here by giving a new LP relaxation and a rounding algorithm for it, and then giving a high-level overview of the analysis. In Section 2.7 we then give more details of the analysis.

2.6.1 LP Relaxation

We now design algorithms for approximating f-FAULT-TOLERANT k-SPANNER when $k \in \{3,4\}$ in both the directed and undirected setting. As in the previous section, we will mostly consider the directed setting since our bounds there immediately imply the same bounds for undirected graphs. Our goal is to give approximation algorithms with performance that does not degrade with f, and in particular to prove Theorem 2.1.3. The $k=3$ case is a simplified version of the $k=4$ case, so in order to be more general we will focus on $k=4$ for most of what remains.

There is a natural LP relaxation for f-FAULT-TOLERANT k-SPANNER which (informally) is just the basic flow LP (2.1) but where the ability to send one unit of flow must hold in $G \setminus F$ for all fault sets F. This LP has been used in approximations for f-FAULT-TOLERANT k-SPANNER (Dinitz and Krauthgamer, 2011a), but when rounding this LP we need to take a union bound over all

$$
\begin{aligned}
\min \quad & \sum_{e \in E} x_e & & \\
\text{s.t.} \quad & \sum_{P \in \mathcal{P}_{s,t} : e \in P} f_P \leq x_e & & \forall (s,t) \in E, \forall e \in E \\
& (f+1)x_e + \sum_{P \in \mathcal{P}_{s,t}} f_P \geq f+1 & & \forall e = (s,t) \in E \\
& \sum_{P \in \mathcal{P}_{s,t} : v \in P} f_P \leq 1 & & \forall (s,t) \in E, \forall v \in V \backslash \{s,t\} \\
& 0 \leq x_e \leq 1 & & \forall e \in E \\
& f_P \geq 0 & & \forall (s,t) \in E, P \in \mathcal{P}_{s,t}
\end{aligned}
\tag{2.8}
$$

Figure 2.2: LP relaxation for f-FAULT-TOLERANT k-SPANNER

fault sets, and hence some dependence on f is clearly necessary. To get around this, we use a different relaxation which is an adaptation of a relaxation first introduced for the special case of $k = 2$ (Dinitz and Krauthgamer, 2011b). Our new LP is given in Figure 2.2, and will henceforth be referred to as LP (2.8).

Intuitively, the LP requires for every edge (s,t) that either (s,t) itself is (fractionally) in the spanner, or that there are $f+1$ node-disjoint paths from s to t with stretch at most k in the spanner. Clearly since k is constant we can solve this in polynomial time. Note that it is not obvious that this is a valid relaxation: the existence of $f+1$ disjoint low-stretch paths clearly guarantees that the spanner is f-fault-tolerant, but it is not clear that this is a necessary condition. And, in fact, it is not: when $k \geq 5$ there are examples of graphs with nodes s, t such that it is only possible to construct one disjoint path of length at most 5 from s to t, but it takes two node deletions to make their distance larger than 5 (see Lovasz, Neumann-Lara, and Plummer, 1978, Figure 1). In other words, the fault-tolerance can be larger than the number of short disjoint paths.

On the other hand, (Lovasz, Neumann-Lara, and Plummer, 1978) proved that for $k = 2, 3, 4$, this situation is not possible. For paths of these very short lengths, the maximum number of node-disjoint paths of length at most k is exactly equal to the minimum number of nodes necessary to remove to make the distance larger than k. Hence LP (2.8) is indeed a valid relaxation for $k \in \{3, 4\}$.

Clearly LP (2.8) is at least as strong as the natural relaxation. Intuitively, though, it is much stronger: for fault sets with size $(1 - \varepsilon)f$, there is not just 1 unit of flow from s to t, but in fact there are εf units of flow. This will enable us to round in a way that is independent of f, and yet still get bounds that hold with high enough probability that we can do a union bound over (almost) all fault sets. The only issue with this approach is if we try to consider fault sets which have size f (or very close to f), since for these sets (as in the natural relaxation) there will only be constant flow, so we cannot get high enough probability bounds. This is why we give only bicriteria approximations: by giving up on extremely large fault sets, we can union bound over the rest.

2.6.2 Rounding Algorithm and Cost Analysis

Our rounding algorithm is essentially the same as Algorithm 1, but with slightly different parameters (namely, $\beta = \sqrt{n}$ rather than $n^{1/3}$, and with some extra polylogarithmic losses) and without the Poisson sampling. We begin our algorithm by solving LP (2.8). Let x_e^* and f_P^* be the fractional optimal solution for all $e \in E$ and paths P. We then round this solution using

Algorithm 3, which returns an edge set E_H. Note that unlike the non-fault-tolerant case, we cannot use tree or arborescence sampling to handle thick edges, and hence our rounding algorithm needs to work for *all* edges.

Algorithm 3 Fault-Tolerant Rounding

$\beta \leftarrow \sqrt{n}$
$C \leftarrow 10^{13}$
$\alpha \leftarrow \frac{C\beta \ln^{26} n}{\varepsilon^3}$
$E_H \leftarrow \varnothing$
for each edge $e \in E$ **do**
    $\hat{x}_e \leftarrow \frac{\lceil x_e^* \cdot n \rceil}{n}$
end for
for each vertex $v \in V$ **do**
    $r_v \leftarrow$ uniform sample from $[0, 1]$
end for
for each edge $e = (s, t) \in E$ **do**
    if $r_s \cdot r_t \leq \alpha \hat{x}_e$ **then**
        add e to E_H
    end if
end for
return E_H

It is easy to see that the cost analysis of Section 2.3.1 still goes through but in an even simpler way, since there is no longer any Poisson rounding. Indeed, it is straightforward to see that $\sum_e \hat{x}_e \leq \sum_e x_e^* + n^2 \times \frac{1}{n} \leq 2OPT(f)$. Now the same analysis as in Lemma 2.3.3 implies that $\mathbb{E}[|E_H|] = O(\alpha \ln n \cdot \sum_e \hat{x}_e)$. Putting this together, we have $\mathbb{E}[|E_H|] = O(\alpha \ln n \cdot OPT(f))$.

We leave the formal proof of Theorem 2.1.3 to Section 2.7, but give some brief intuition here. Consider an edge $(s, t) \in E$. Similarly to the non-fault-tolerant case, we say that E_H *settles* (s, t) if for all fault sets $F \subseteq V \setminus \{s, t\}$ with $|F| \leq (1 - \epsilon)f$, there is a path of length at most 4 from s to t in $G \setminus F$ that uses only edges in E_H. If significant flow is sent along (s, t) itself then it is

straightforward to argue that E_H settles (s, t). Otherwise, consider a fault set $F \subseteq V \setminus \{s, t\}$ with $|F| \leq (1 - \epsilon)f$. Then in $G \setminus F$ there is still at least ϵf flow along paths of length 2, 3 or 4 from s to t. The most difficult case is if most of the remaining flow is along paths of length 4 (although the length 3 case is not at all trivial), so assume there is at least $\frac{\epsilon}{3}f$ flow on paths of length 4.

As in the non-fault tolerant case, we can lose polylogarithmic factors in the flow and bucket the paths so that certain parameters (total flow along second hop, total flow along third hop, load of first node, and load of last node) are essentially identical for each path we consider. We can then break into what are essentially much more complicated versions of the same cases as in the non-fault tolerant setting.

As an example, recall that in the non-fault tolerant setting the simplest case was when one node in the middle layer (S_2) had large load, in which case we could directly analyze the probability that this node was the center of a 4-path from s to t. In the fault-tolerant case, we need this to hold with such high probability (essentially $1 - \frac{1}{n^f}$) that we cannot depend on a single node having large load. We instead need to define the case to be where a significant fraction of the flow is sent through *a set* of nodes with large load. And we can no longer directly analyze this, instead needing a much more complicated analysis. We cannot even use Lemma 2.2.4 – we need a different way of analyzing this case.

In this case we first set thresholds p_1, p_2, p_3 such that if $r_{u_P} \leq p_1$ and $r_{v_P} \leq p_2$ and $r_{w_P} \leq p_3$ then E_H will include the entire path P. If a node v has r_v less than its threshold, then we say that v is *selected*. We then flip the

random coins for S_1, and show that with extremely high probability there are many nodes in S_2 that are adjacent to at least one selected node in S_1. Call these nodes *partially selected*. This is actually a nontrivial lemma (Lemma 2.7.10), since for two nodes in S_2 the event of being partially selected is not independent if their neighborhoods are overlapping. So instead of using a Chernoff bound, we have to rewrite the process as a martingale and use Azuma's inequality.

Once we know that there are many partially selected nodes in S_2, we can repeat our analysis for S_3 to show that with extremely high probability there are many partially selected nodes in S_2 that are also adjacent to a selected node in S_3. Hence there are many nodes in S_2 who have selected neighbors in both S_1 and S_3, so we can finally flip the random coins for S_2 to prove that with extremely high probability there is some path in which all nodes are selected.

All other cases are handled in either a similar way, through the use of Lemma 2.2.4 to directly prove the existence of many disjoint paths, or by using another method to prove concentration despite correlations (e.g., Janson's inequality).

2.7 Details of f-FAULT-TOLERANT k-SPANNER

Recall the definition of an edge set settling an edge:

Definition 2.7.1. A set $E' \subseteq E$ "settles" an edge $(s,t) \in E$ if for all sets $F \subseteq V \setminus \{s,t\}$ with $|F| \leq (1-\varepsilon)f$ there is a path from s to t in $G \setminus F$ of length at most k that only uses edges in E'.

Now we start to show that Algorithm 3 returns a $(1-\varepsilon)f$-fault tolerant spanner with probability at least $\frac{1}{n}$, which together with the previous cost analysis suffices to prove Theorem 2.1.3.

Depending on how the flow splits in the optimal solution of linear program (2.8), we can use different approaches to prove that with high probability, there is a spanning path remaining after faults. We first introduce a definition which will let us define some cases of interest and help analyze others.

Definition 2.7.2 (($s, t, c_1, \ldots, c_k, l_2, \ldots, l_{k-2}, F, f$)-scene)**.** Given two nodes $s, t \subseteq V$ and a fault set $F \subseteq V \backslash \{s, t\}$, an $(s, t, c_1, \ldots, c_k, l_2, \ldots, l_{k-2}, F, f)$-scene is a $(k+1)$-layer graph $(S_0 = \{s\}, S_1, \ldots, S_{k-1}, S_k = \{t\}, E')$ which is a subgraph of $G \backslash F$ and:

1. For $i \in [k]$, each edge e that goes from S_{i-1} to S_i has $\hat{x}_e \geq c_i$.

2. If we let the capacity of each edge e that goes from S_{i-1} to S_i be $2c_i$, $i = 1, \ldots, k$, and also let the capacity of each node in S_i be l_i, $i = 2, \ldots, k-2$, then the max flow from s to t is at least f.

Note that in this definition, if $k = 3$ then there are no l values, while if $k = 4$ there is simply l_2. We now use this definition to prove two lemmas, each of which corresponds to a case which can be easily handled using Lemma 2.2.4. The first lemma gives a sufficient condition for E_H to contain at least $f+1$ internally vertex disjoint paths of length 3 from s to t (and thus a sufficient condition for E_H to settle (s, t)).

Lemma 2.7.3. *For each $(s, t) \in E$, if there is a $(s, t, a, c, b, \varnothing, f')$-scene with $f' \geq \frac{\varepsilon(f+198)}{24\lceil \log n \rceil^3}$, $\frac{1}{n} \leq c \leq a \leq \frac{1}{\alpha}$, and $c \leq b \leq \frac{1}{\alpha}$, then E_H contains at least $f+1$ length 3*

disjoint paths from s to t with probability at least $1 - \frac{1}{n^3}$.

Proof. Let G' be this $(s, t, a, c, b, \varnothing, f')$-scene. Let B be all the possible paths from s to t in G'. Each $P \in B$ has form $s \to u_P \to v_P \to t$. Note that if

$$r_{u_P} \leq \min\left\{\frac{c}{b}, a\alpha\right\} \text{ and } r_{v_P} \leq b\alpha \tag{2.9}$$

then all edges of P are in E_H. This is because if (2.9) holds, then $r_s r_{u_P} \leq r_{u_P} \leq a\alpha \leq \hat{x}_{(s,u_P)}\alpha$ and so $(s, u_P) \in E_H$. Similarly, at the other end of the path we would have that $r_{v_P} r_t \leq r_{v_P} \leq b\alpha \leq \hat{x}_{(v_P,t)}\alpha$, so $(v_P, t) \in E_H$. For the middle edge of the path, we would have that $r_{u_P} r_{v_P} \leq \frac{c}{b} \cdot b\alpha = c\alpha \leq \hat{x}_{(u_P,v_P)}\alpha$, so $(u_P, v_P) \in E_H$.

We can now use Lemma 2.2.4, treating S_2 as layer 1 and S_1 as layer 2 in the lemma. Then using the notation of Lemma 2.2.4, we know that $c_1 = 2b$ and $c_2 = 2a$. Let $p_1 = b\alpha$, $p_2 = \min\{\frac{c}{b}, a\alpha\}$. Then, in order to prove that there exists $f + 1$ disjoint good paths $P \in E_H$ with high probability, we only need to prove $(f+1) \cdot \sum_{i=1}^{2} \left(c_i \cdot \prod_{j=i}^{2} \lceil \frac{4 \ln n}{p_j} \rceil \right) < f'$. Because $a \in [\frac{1}{n}, \frac{1}{\alpha})$, $b \in [\frac{1}{n}, \frac{1}{\alpha})$, $c \geq \frac{1}{n}$,

and $p_1 < 1, p_2 < 1$, we have that

$$
\begin{aligned}
&(f+1)\cdot\sum_{i=1}^{2}\left(c_i\cdot\prod_{j=i}^{2}\left\lceil\frac{4\ln n}{p_j}\right\rceil\right)\\
&<(f+1)\cdot\left(2b\cdot\frac{5\ln n}{b\alpha}\cdot\frac{5\ln n}{\min\{\frac{c}{b},a\alpha\}}+2a\cdot\frac{5\ln n}{\min\{\frac{c}{b},a\alpha\}}\right)\\
&=(f+1)\cdot\left(\frac{50\ln^2 n}{\alpha\min\{\frac{c}{b},a\alpha\}}+\frac{10\ln n}{\min\{\frac{c}{ab},\alpha\}}\right)\\
&\le(f+1)\cdot\left(\frac{50\ln^2 n}{\alpha\min\{\frac{1/n}{1/\alpha},\frac{1}{n}\cdot\alpha\}}+\frac{10\ln n}{\min\{\frac{1/n}{\frac{1}{\alpha}\cdot\frac{1}{\alpha}},\alpha\}}\right)\\
&\le(f+1)\cdot(50\ln^2 n+10\ln n)\cdot\frac{n}{(\frac{C\sqrt{n}\ln^{26} n}{\varepsilon^3})^2}\\
&<\frac{\varepsilon(f+198)}{24\lceil\log n\rceil^3}=f'
\end{aligned}
$$

Therefore, with probability at least $1-\frac{1}{n^3}$, E_H contains $f+1$ disjoint $s-t$ paths with length 3. □

The next lemma is similar, but for paths of length 4.

Lemma 2.7.4. *For each $(s,t)\in E$, if there is a $(s,t,a,c,d,b,\frac{1}{\beta},\varnothing,f')$-scene with $f'\ge\frac{\varepsilon(f+198)}{324\lceil\log n\rceil^4}$, $\frac{1}{n}\le c\le a\le\frac{1}{\alpha}$, $\frac{1}{n}\le d\le b\le\frac{1}{\alpha}$, then with probability at least $1-\frac{1}{n^3}$, E_H contains at least $f+1$ disjoint paths from s to t of length 4.*

Proof. Without loss of generality, we can assume that $\frac{c}{a}\le\frac{d}{b}$ (since otherwise we can simply consider the graph from the reverse direction). Let G' be this $(s,t,a,c,d,b,\frac{1}{\beta},\varnothing,f')$-scene. Let B be all the possible paths from s to t in G'.

Each $P \in B$ has the form $s \to u_P \to v_P \to w_P \to t$. Note that if

$$r_{u_P} \leq a\alpha \text{ and } r_{w_P} \leq b\alpha \text{ and } r_{v_P} \leq \frac{c}{a} \tag{2.10}$$

then all edges of P are in E_H. This is easy to see by simple calculations: if (2.10) holds, then $r_s r_{u_P} \leq r_{u_P} \leq a\alpha \leq \hat{x}_{(s,u_P)}\alpha$ and so $(s, u_P) \in E_H$. Similarly, at the other end of the path we would have that $r_{w_P} r_t \leq r_{w_P} \leq b\alpha \leq \hat{x}_{(w_P,t)}\alpha$, so $(w_P, t) \in E_H$. For the second edge of the path, we would have that $r_{u_P} r_{v_P} \leq a\alpha \cdot \frac{c}{a} = c\alpha \leq \hat{x}_{(u_P,v_P)}\alpha$, so $(u_P, v_P) \in E_H$. Finally, for the third path edge we would have that $r_{v_P} r_{w_P} \leq \frac{c}{a} \cdot b\alpha \leq \frac{d}{b} \cdot b\alpha = d\alpha \leq \hat{x}_{(v_P,w_P)}\alpha$, and so $(v_P, w_P) \in E_H$.

We can now use Lemma 2.2.4, treating S_3 as layer 1, S_2 as layer 2, and S_1 as layer 3 in the lemma. Using the notation of Lemma 2.2.4, we know that $c_1 = 2b$, $c_2 = \frac{1}{\beta}$, and $c_3 = 2a$. Let $p_1 = b\alpha$, $p_2 = \frac{c}{a}$, $p_3 = a\alpha$. Then, in order to prove that there exists $f + 1$ disjoint good paths $P \in E_H$ with high probability, we only need to prove $(f+1) \cdot \sum_{i=1}^{3} \left(c_i \cdot \prod_{j=i}^{3} \lceil \frac{4 \ln n}{p_j} \rceil \right) < f'$. Because

$c \geq \frac{1}{n}$, $d \geq \frac{1}{n}$, and $p_1 < 1, p_2 < 1, p_3 < 1$, we have that

$$
\begin{aligned}
&(f+1) \cdot \sum_{i=1}^{3} \left(c_i \cdot \prod_{j=i}^{3} \left\lceil \frac{4 \ln n}{p_j} \right\rceil \right) \\
&< (f+1) \cdot \left(2b \cdot \frac{5 \ln n}{b\alpha} \cdot \frac{5 \ln n}{\frac{c}{a}} \cdot \frac{5 \ln n}{a\alpha} + \frac{1}{\beta} \cdot \frac{5 \ln n}{\frac{c}{a}} \cdot \frac{5 \ln n}{a\alpha} + 2a \cdot \frac{5 \ln n}{a\alpha} \right) \\
&= (f+1) \cdot \left(\frac{250 \ln^3 n}{c\alpha^2} + \frac{25 \ln^2 n}{d\alpha\beta} + \frac{10 \ln n}{\alpha} \right) \\
&\leq (f+1) \cdot \left(\frac{250 \ln^3 n}{\frac{1}{n}\alpha^2} + \frac{25 \ln^2 n}{\frac{1}{n}\alpha\beta} + \frac{10 \ln n}{\alpha} \right) \\
&\leq (f+1) \cdot \left(\frac{250 n \ln^3 n}{(\frac{C\sqrt{n} \ln^{26} n}{\varepsilon^3})^2} + \frac{25 n \ln^2 n}{\frac{C\sqrt{n} \ln^{26} n}{\varepsilon^3} \cdot \sqrt{n}} + \frac{10 \ln n}{\frac{C\sqrt{n} \ln^{26} n}{\varepsilon^3}} \right) \\
&< \frac{\varepsilon(f+198)}{324 \lceil \log n \rceil^4} = f'
\end{aligned}
$$

Therefore, with probability at least $1 - \frac{1}{n^3}$, E_H contains $f+1$ length 4 disjoint paths from s to t. □

These two lemmas let us directly prove the existence of many vertex-disjoint paths (and thus fault-tolerance). In other situations, though, we need to reason about possible fault sets, in which case the following definition is useful.

Definition 2.7.5 (remaining edge set $E_H \backslash F$)**.** Given a graph $G = (V, E)$, an edge set $E_H \subseteq E$, and a fault set $F \subseteq V$, let $E_H \backslash F$ be the set of edges in E_H which do not have an endpoint in F.

Now we can finally prove the main fault-tolerant theorem.

2.7.1 Proof Sketch of Theorem 2.1.3

We want to prove that for any edge $(s,t) \in E$, E_H settles (s,t) with probability at least $1 - \frac{1}{n^3}$. After that, because there are at most n^2 edges in the graph, we can use a union bound over all edges to finish the proof of the theorem.

From the linear program (2.8) we know that either $\sum_{P \in \mathcal{P}_{s,t}} f_P^* \geq (1 - \frac{1}{\alpha})(f+1)$, or $x^*_{(s,t)} \geq \frac{1}{\alpha}$.

Note that if $x^*_{(s,t)} \geq \frac{1}{\alpha}$, then $\hat{x}_{(s,t)} \geq \frac{1}{\alpha}$, so $(s,t) \in E_H$ with probability 1 and hence E_H settles (s,t). So for the rest of the proof we will assume $\sum_{P \in \mathcal{P}_{s,t}} f_P^* \geq (1 - \frac{1}{\alpha})(f+1)$.

If $k = 3$ and there is a scene which satisfies Lemma 2.7.3, then E_H contains $f+1$ disjoint paths from s to t of length at most 3. If $k = 4$ and there is a scene which satisfies Lemma 2.7.3 or Lemma 2.7.4, then E_H contains $f+1$ disjoint paths from s to t of length at most 4. These implies that E_H settles (s,t). So for the rest of the proof we will also assume there is no scene which satisfies Lemma 2.7.3, and there is no scene which satisfies Lemma 2.7.4 when $k = 4$.

In order to prove that E_H settles (s,t) with probability at least $1 - \frac{1}{n^3}$, we only need to prove that for each fault set $F \subseteq V$ with $|F| \leq (1-\varepsilon)f$, the probability that there is no path of length at most k in $E_H \backslash F$ from s to t is at most $\frac{1}{n^{f+3}}$. A union bound over all possible fault sets (of which there are at most $\binom{n}{f} \leq n^f$) then gives the result.

Consider a fault set $F \subseteq V \setminus \{s,t\}$ with $|F| \leq (1-\varepsilon)f$. If $\varepsilon < \frac{1}{200}$ is a small

constant then $\varepsilon \gg \frac{1}{\alpha}$, and so from the linear program (2.8) we know that

$$\begin{aligned}\sum_{P\in\mathcal{P}_{s,t},P\cap F=\varnothing} f_P^* &\geq \sum_{P\in\mathcal{P}_{s,t}} f_P^* - \sum_{v\in F}\sum_{P\in\mathcal{P}_{s,t},v\in P} f_P^* \\ &\geq \left(1-\frac{1}{\alpha}\right)(f+1) - (1-\varepsilon)f\cdot 1 \\ &\geq \frac{\varepsilon}{2}(f+198)\end{aligned}$$

For $i \in \{2,3,4\}$, let $\mathcal{P}_{s,t,F}^{(i)}$ denote all the paths in $\mathcal{P}_{s,t}$ that have length exactly i and do not intersect with F. We know that one of these three sets must have $\frac{\varepsilon(f+198)}{6}$ flow because the total flow is at least $\frac{\varepsilon}{2}(f+198)$. This naturally gives rise to three cases. We will consider all these cases in the following lemmas: Lemma 2.7.6 for length 2 paths, Lemma 2.7.14 for length 3 paths, and Lemma 2.7.15 for length 4 paths. We show that in each of these cases, there is a short path in $E_H\backslash F$ from s to t with probability at least $1-\frac{1}{n^{f+3}}$. Combining these lemmas and taking appropriate union bounds completes the proof.

2.7.2 Proving Paths Exist with High Probability

We now need to prove the three mentioned lemmas.

Lemma 2.7.6. *For each $(s,t) \in E$ and $F \subseteq V\setminus\{s,t\}$, if $\sum_{P\in\mathcal{P}_{s,t,F}^{(2)}} f_P^* \geq \frac{\varepsilon(f+198)}{6}$, then with probability at least $1-\frac{1}{n^{f+3}}$, there is a path in $E_H\backslash F$ from s to t with length 2.*

Proof. Every length 2 path $P \in \mathcal{P}_{s,t,F}^{(2)}$ must have the form $s \to v_P \to t$ and all v_P must be different. If there exists a path P such that $\hat{x}_{(s,v_P)} \geq \frac{1}{\alpha}$ and

$\hat{x}_{(v_P,t)} \geq \frac{1}{\alpha}$, then $(s, v_P) \in E_H \backslash F$ and $(v_P, t) \in E_H \backslash F$, so with probability 1, there is a path in $E_H \backslash F$ from s to t with length 2.

Otherwise, if there does not exist a path P such that $\hat{x}_{(s,v_P)} \geq \frac{1}{\alpha}$ and $\hat{x}_{(v_P,t)} \geq \frac{1}{\alpha}$, then $f_P^* < \frac{1}{\alpha}$ for all $P \in \mathcal{P}^{(2)}_{s,t,F}$, so there are at least $\frac{\frac{\varepsilon(f+198)}{6}}{\frac{1}{\alpha}} = \frac{\varepsilon\alpha(f+198)}{6}$ such paths. We also know that for each path $P \in \mathcal{P}^{(2)}_{s,t,F}$, if $r_{v_P} \leq \frac{\alpha}{n}$, then (s, v_P) and (v_P, t) must be in $E_H \backslash F$ because $\hat{x}_{(s,v_P)} \geq \frac{1}{n}$ and $\hat{x}_{(v_P,t)} \geq \frac{1}{n}$. The probability that there is no $r_{v_P} \leq \frac{\alpha}{n}$ is at most $(1 - \frac{\alpha}{n})^{\frac{\varepsilon\alpha(f+198)}{6}} \leq e^{-\frac{\varepsilon\alpha^2(f+198)}{6n}} \leq \frac{1}{n^{f+3}}$ because $\alpha = \frac{C\beta \ln^{26} n}{\varepsilon^3}$ and $\beta = \sqrt{n}$. Thus with probability at least $1 - \frac{1}{n^{f+3}}$, there is a path in $E_H \backslash F$ from s to t with length 2. □

The proofs of the other two lemmas both proceed in roughly the same way. We first bucket the paths into appropriate scenes, arguing that since there are not many possible scenes there must be one bucket with a polylogarithmic amount of flow. We fix this bucket, and break into cases depending on how the parameters of the scene relate to each other. For each case, we prove that with appropriately high probability, the claimed path exists in the bucket. The exact proof method is highly dependent on the case, ranging from relatively straightforward direct analysis to more complicated cases in which we resort to framing the number of intermediate "selected" nodes as a martingale and using Azuma's inequality to guarantee concentration.

Lemma 2.7.7 (Janson's inequality)**.** *Let $R = R_{p_1,\ldots,p_n}$ be a random subset of $[n]$ formed by including each $i \in [n]$ in R with probability p_i, independently. Let $\mathcal{S}$ be a family of subset of $[n]$, and for each $A \in \mathcal{S}$, let $X_A = \mathbb{1}_{A \subseteq R}$, $X = \sum_{A \in \mathcal{S}} X_A$ and*

$\Delta = \sum_{A,B \in \mathcal{S}: A \cap B \neq \varnothing} \mathbf{Pr}[X_A = X_B = 1]$, $t \in [0, \mathbb{E}[X]]$. *Then*

$$\mathbf{Pr}[X \leq \mathbb{E}[X] - t] \leq e^{-\frac{t^2}{2\mathbb{E}[X]+\Delta}}$$

Definition 2.7.8 (martingale)**.** A sequence of random variables $X_0, X_1, \ldots$ is a *martingale* if for each $i \in \mathbb{N}$, $\mathbb{E}[|X_i|] < \infty$ and $\mathbb{E}[X_{i+1} - X_i \mid X_0, \ldots, X_i] = 0$.

Lemma 2.7.9 (Azuma's inequality)**.** *If a sequence of random variables $X_0, X_1, \ldots$ is a martingale and $|X_i - X_{i-1}| < c_i$ for all i, then*

$$\mathbf{Pr}[|X_n - X_0| \geq t] \leq 2e^{-\frac{t^2}{2\sum_{i=1}^{n} c_i^2}}$$

In particular, we sometimes argue if we have a bipartite graph with particular parameters (as we do between any two layers of a scene), then picking nodes independently from one side results in many neighbors on the other side, despite correlations induced by the edge structure. By considering each layer of the scene appropriately, this allows us to bound the existence of paths. This was the notion of *partially selecting* nodes that was discussed in Section 2.6.

To be more specific, we need the following lemma. For any vertex set S, we define $N(S)$ as the neighbor of set S. The lemma shows that given a bipartite graph where both sides have bounded degree, if we randomly choose each node on one side with a large enough probability, then the chosen set will have a large neighborhood with very high probability.

Lemma 2.7.10. *Let (L, R, E) be a bipartite graph. Each node in L has degree at most d_L and the average degree of nodes in L is at least $\frac{d_L}{r_L}$, and each node in R has degree at most d_R and the average degree of nodes in R is at least $\frac{d_R}{r_R}$. If we pick each node*

in L independently with probability $p \geq \frac{32r_L^2}{d_R}$ to get a random subset $S \subseteq L$, then $\mathbf{Pr}\left[|N(S)| < \frac{|E|}{2d_R}\right] \leq 2e^{-\frac{|E|p}{256d_L r_L^2}}$*. (In other words, with very high probability at least a $\frac{1}{2r_R}$ fraction of the nodes in R are adjacent to a node in S).*

Proof. We will sometimes refer to L as the "left" and R as the "right". The total number of nodes on the left is at least $\frac{|E|}{d_L}$, and we select each node with probability p. Thus by a Chernoff bound, $\mathbf{Pr}\left[|S| < \frac{|E|p}{2d_L}\right] \leq e^{-\frac{|E|p}{8d_L}}$. Hence, we can assume we always have $|S| \geq \frac{|E|p}{2d_L}$ by taking a union bound in the end with the probability that this is false. Now, we can treat this selection of nodes as a procedure that iteratively selects nodes uniformly in L without replacement, until we get $\frac{|E|p}{2d_L}$ nodes.

Let $v_i \in L$ be the node selected in iteration i. Then in each iteration i, we will delete not only v_i, but also all of the nodes in $N(v_i) \subseteq R$. So in each iteration, $|L|$ decreases by 1 and $|R|$ decreases by a variable amount. Let t be the first iteration in which $|R| \leq \frac{|E|}{2d_R}$ at the beginning of the iteration.

For $i = 1, \ldots, \frac{|E|p}{2d_L}$, we define some new random variables: $Z_i = 1$ with probability $\frac{1}{4r_L}$, and 0 otherwise. These random variables are independent of all other random variables. Let $X_i = \mathbb{1}_{i \geq t} \cdot Z_i + \mathbb{1}_{i<t} \cdot \mathbb{1}_{degree(v_i) \geq \frac{d_L}{4r_L}}$ where $\mathbb{1}$ denotes an indicator variable.

We know that $X_i \in \{0, 1\}$. By definition, $X_i = 1$ only when $i \geq t$ and $Z_i = 1$, or $i < t$ and $degree(v_i) \geq \frac{d_L}{4r_L}$.

Claim 2.7.11. $\mathbb{E}[X_i \mid X_1, \ldots, X_{i-1}] \geq \frac{1}{4r_L}$ *for all i.*

Proof. First, if $i \geq t$ then $X_i = Z_i$ and so is independent of $X_1, \ldots, X_{i-1}$. Hence in this case $\mathbb{E}[X_i \mid X_1, \ldots, X_{i-1}] = \frac{1}{4r_L}$.

Alternatively, if $i < t$, then total edges remaining is at least $|E| - \frac{|E|}{2d_R} \cdot d_R = \frac{|E|}{2}$. Assuming $\mathbf{Pr}\left[degree(v_i) \geq \frac{d_L}{4r_L} \mid X_1, \ldots, X_{i-1}\right] < \frac{1}{4r_L}$, then the total number of edges is at most $|L| \cdot \frac{1}{4r_L} \cdot d_L + |L| \cdot (1 - \frac{1}{4r_L}) \cdot \frac{d_L}{4r_L} < \frac{|E|}{2}$ (where the final inequality is because $|L| \leq \frac{|E|}{d_L/r_L}$). This gives a contradiction. Therefore $\mathbf{Pr}\left[degree(v_i) \geq \frac{d_L}{4r_L} \mid X_1, \ldots, X_{i-1}\right] \geq \frac{1}{4r_L}$, and thus $\mathbb{E}[X_i \mid X_1, \ldots, X_{i-1}] \geq \frac{1}{4r_L}$ regardless of i. □

Claim 2.7.12. $\mathbf{Pr}\left[|N(S)| < \frac{|E|}{2d_R}\right] \leq \mathbf{Pr}\left[\sum_{i=1}^{\frac{|E|p}{2d_L}} X_i \leq \frac{2|E|r_L}{d_L d_R}\right]$

Proof. Because if $|N(S)| < \frac{|E|}{2d_R}$, then $t > \frac{|E|p}{2d_L}$, which is the total number of the iterations. So we only need to prove that if $t > \frac{|E|p}{2d_L}$, then $\sum_{i=1}^{\frac{|E|p}{2d_L}} X_i \leq \frac{2|E|r_L}{d_L d_R}$.

If $t > \frac{|E|p}{2d_L}$, assume $\sum_{i=1}^{\frac{|E|p}{2d_L}} X_i > \frac{2|E|r_L}{d_L d_R}$, then because $X_i = \mathbb{1}_{degree(v_i) \geq \frac{d_L}{4r_L}}$, the total number of nodes which is deleted on the right in $\frac{|E|p}{2d_L}$ iterations is $> \frac{d_L}{4r_L} \cdot \frac{2|E|r_L}{d_L d_R} = \frac{|E|}{2d_R}$, which means $t \leq \frac{|E|p}{2d_L}$, contradicted. Therefore $\sum_{i=1}^{\frac{|E|p}{2d_L}} X_i \leq \frac{2|E|r_L}{d_L d_R}$, thus we have proved the lemma. □

Let $Y_i = \sum_{j=1}^{i} (X_i - \mathbb{E}[X_i \mid X_1, \ldots, X_{i-1}])$ for all i and $Y_0 = 0$.

Claim 2.7.13. *$Y_0, \ldots, Y_{\frac{|E|p}{2d_L}}$ is a martingale.*

Proof. For all i, we have

$$\mathbb{E}[Y_{i+1} - Y_i \mid X_1, \ldots, X_i] = \mathbb{E}[X_{i+1} - \mathbb{E}[X_{i+1} \mid X_1, \ldots, X_i] \mid X_1, \ldots, X_i] = 0$$

Because $Y_0, \ldots, Y_i$ are functions of $X_1, \ldots, X_i$, therefore for all i, $\mathbb{E}[Y_{i+1} - Y_i \mid Y_0, \ldots, Y_i] = 0$, thus $Y_0, \ldots, Y_{\frac{|E|p}{2d_L}}$ is a martingale. □

We also know that $|Y_i - Y_{i-1}| = |X_i - \mathbb{E}[X_i \mid X_1, \ldots, X_{i-1}]| \leq 1$ because $X_i \in \{0, 1\}$. Hence by Azuma's inequality (Lemma 2.7.9), we have

$$\mathbf{Pr}\left[\sum_{i=1}^{\frac{|E|p}{2d_L}} X_i \leq \frac{2|E|r_L}{d_L d_R}\right] = \mathbf{Pr}\left[Y_{\frac{|E|p}{2d_L}} \leq \frac{2|E|r_L}{d_L d_R} - \sum_{i=1}^{\frac{|E|p}{2d_L}} \mathbb{E}[X_i \mid X_1, \ldots, X_{i-1}]\right] \tag{2.11}$$

$$\leq \mathbf{Pr}\left[Y_{\frac{|E|p}{2d_L}} \leq \frac{2|E|r_L}{d_L d_R} - \frac{|E|p}{2d_L} \cdot \frac{1}{4r_L}\right] \tag{2.12}$$

$$\leq \mathbf{Pr}\left[Y_{\frac{|E|p}{2d_L}} \leq -\frac{|E|p}{16 d_L r_L}\right] \tag{2.13}$$

$$\leq 2e^{-\frac{|E|p}{256 d_L r_L^2}} \tag{2.14}$$

Equation (2.11) is because of the definition of $Y_{\frac{|E|p}{2d_L}}$, inequality (2.12) is because of Claim 2.7.11, inequality (2.13) is because $p \geq \frac{32 r_L^2}{d_R}$, and inequality (2.14) is because of the Azuma's inequality and $|E| \geq N$.

Therefore from claim 2.7.12 and a union bound, we know that

$$\mathbf{Pr}\left[N(S) < \frac{|E|}{2d_R}\right] \leq e^{-\frac{|E|p}{256 d_L r_L^2}} + e^{-\frac{|E|p}{8 d_L}} \leq 2e^{-\frac{|E|p}{256 d_L r_L^2}}.$$

□

This is the key lemma which allows us to prove the next two lemmas, and thus Theorem 2.1.3.

Lemma 2.7.14. *For each $(s,t) \in E$ and $F \subseteq V \setminus \{s,t\}$, if $\sum_{P \in \mathcal{P}^{(3)}_{s,t,F}} f^*_P \geq \frac{\varepsilon(f+198)}{6}$ and there is no scene which satisfies Lemma 2.7.3, then with probability at least $1 - \frac{1}{n^{f+3}}$, there is a path in $E_H \backslash F$ from s to t with length 3.*

Proof. We begin by randomly partitioning $V_{s,t} \backslash (\{s,t\} \cup F)$ to 2 parts S_1, S_2. Then the probability that each node is in partition S_j is $\frac{1}{2}$ for $j \in \{1,2\}$. Let $\mathcal{P}' \subseteq \mathcal{P}^{(3)}_{s,t,F}$ be the collection of paths of the form $s \to u \to v \to t$ where $u \in S_1$, $v \in S_2$. Clearly each path $P \in \mathcal{P}^{(3)}_{s,t,F}$ is in $\mathcal{P}'$ with probability $(\frac{1}{2})^2 = \frac{1}{4}$. So $\mathbb{E}[\sum_{P \in \mathcal{P}'} f^*_P] \geq \frac{\varepsilon(f+198)}{6} \cdot \frac{1}{4} = \frac{\varepsilon(f+198)}{24}$, and hence there is *some* partition S_1, S_2 where $\sum_{P \in \mathcal{P}'} f^*_P \geq \frac{\varepsilon(f+198)}{24} \geq \varepsilon$. Fix this partition and collection of paths $\mathcal{P}'$, and let $f' = \sum_{P \in \mathcal{P}'} f^*_P$.

For each $e \in E$, let $\hat{f}_e = \sum_{P : P \in \mathcal{P}', e \in P} f^*_P$ be the total flow in $\mathcal{P}'$ that pass through e. For each $P \in \mathcal{P}'$, let u_P be the node of P in S_1, let v_P be the node of P in S_2 (so P is of the form $s \to u_P \to v_P \to t$). We will bucket the paths in $\mathcal{P}'$ according to three values: $\hat{f}_{(s,u_P)}$, $\hat{f}_{(u_P,v_P)}$ and $\hat{f}_{(v_P,t)}$. For each integer $i \in \{1, 2, \ldots, \lceil \log n \rceil - 1\}$, let L_i be the real interval $\left(\frac{2^i}{n}, \frac{2^{i+1}}{n}\right]$. Moreover, let $L_0 = \left(0, \frac{2}{n}\right]$. Then we can construct $\lceil \log n \rceil^3$ buckets as follows.

$$B_{i,j,k} = \left\{ P \in \mathcal{P}' \mid \hat{f}_{(s,u_P)} \in L_i, \hat{f}_{(u_P,v_P)} \in L_j, \text{ and } \hat{f}_{(v_P,t)} \in L_k \right\}$$

There must be one bucket such that $\sum_{P \in B_{i,j,k}} f^*_P \geq \frac{f'}{\lceil \log n \rceil^3}$. To simplify our notation, we call this bucket B. Then there are three values a, b, c, each of

which is a multiple of $\frac{1}{n}$, such that for each $P \in B$, we have

$$\hat{f}_{(s,u_P)} \in \begin{cases} (0, 2a], & \text{if } a = \frac{1}{n} \\ (a, 2a], & otherwise \end{cases},$$

$$\hat{f}_{(u_P,v_P)} \in \begin{cases} (0, 2c], & \text{if } c = \frac{1}{n} \\ (c, 2c], & otherwise \end{cases},$$

$$\hat{f}_{(v_P,t)} \in \begin{cases} (0, 2b], & \text{if } b = \frac{1}{n} \\ (b, 2b], & otherwise \end{cases}.$$

We can easily see that $c \leq a$ and that $c \leq b$, and that the induced subgraph by B is a $(s, t, a, c, b, F, \sum_{P \in B} f_P^*)$-scene.

We further split the problem to 4 cases, depending on whether $a \geq \frac{1}{\alpha}$ and whether $b \geq \frac{1}{\alpha}$.

Case 1: $a < \frac{1}{\alpha}, b < \frac{1}{\alpha}$

This case cannot appear: since $c \leq a < \frac{1}{\alpha}$ and $c \leq b < \frac{1}{\alpha}$, this would imply the existence of a $(s, t, a, c, b, F, \sum_{P \in B} f_P^*)$-scene, which would also be a $(s, t, a, c, b, \varnothing, \sum_{P \in B} f_P^*)$-scene. Such a scene would satisfy the requirement of Lemma 2.7.3, and so by assumption cannot exist.

Case 2: $a \geq \frac{1}{\alpha}, b < \frac{1}{\alpha}$

Let $P \in B$, and note that if

$$r_{v_P} \leq \frac{\alpha}{n} \tag{2.15}$$

then all edges of P are in $E_H \backslash F$. This is because if (2.15) holds, then $r_s r_{u_P} \leq 1 \leq a\alpha \leq \hat{x}_{(s,u_P)}\alpha$ and so $(s, u_P) \in E_H \backslash F$. Similarly, at the other end of the path we would have that $r_{v_P} r_t \leq r_{v_P} \leq \frac{\alpha}{n} \leq \hat{x}_{(v_P,t)}\alpha$, so $(v_P, t) \in E_H \backslash F$. For

the middle edge of the path, we would have that $r_{u_P} r_{v_P} \leq r_{v_P} \leq \frac{\alpha}{n} \leq \hat{x}_{(u_P,v_P)}\alpha$, so $(u_P, v_P) \in E_H \backslash F$.

We know that for each $v_P \in S_2$, there is at most $\hat{f}_{(v_P,t)} \leq 2b \leq \frac{2}{\alpha}$ flow pass through v_P, so the number of different v_P is at least $\frac{\frac{f'}{\lceil \log n \rceil^3}}{\frac{2}{\alpha}} = \frac{\alpha f'}{2\lceil \log n \rceil^3}$. The probability that there is no $r_{v_P} \leq \frac{\alpha}{n}$ is at most $(1 - \frac{\alpha}{n})^{\frac{\alpha f'}{2\lceil \log n \rceil^3}} \leq e^{-\frac{\alpha^2 f'}{2n\lceil \log n \rceil^3}} \leq \frac{1}{n^{f+3}}$.

Thus with probability at least $1 - \frac{1}{n^{f+3}}$, there is a path in $E_H \backslash F$ from s to t with length 3.

Case 3: $a < \frac{1}{\alpha}, b \geq \frac{1}{\alpha}$

This case is symmetric to case 2.

Let $P \in B$, and note that if

$$r_{u_P} \leq \frac{\alpha}{n} \tag{2.16}$$

then all edges of P are in $E_H \backslash F$. The number of different u_P is at least $\frac{\frac{f'}{\lceil \log n \rceil^3}}{\frac{2}{\alpha}} = \frac{\alpha f'}{2\lceil \log n \rceil^3}$. The probability that there is no $r_{u_P} \leq \frac{\alpha}{n}$ is at most $(1 - \frac{\alpha}{n})^{\frac{\alpha f'}{2\lceil \log n \rceil^3}} \leq e^{-\frac{\alpha^2 f'}{2n\lceil \log n \rceil^3}} \leq \frac{1}{n^{f+3}}$.

Thus with probability at least $1 - \frac{1}{n^{f+3}}$, there is a path in $E_H \backslash F$ from s to t with length 3.

Case 4: $a \geq \frac{1}{\alpha}, b \geq \frac{1}{\alpha}$

Consider the graph with node set $\{s, t\} \cup \bigcup_{P \in B} \{v_P, u_P\}$ in this case and let the edge set be $\bigcup_{P \in B} \{(s, u_P), (u_P, v_P), (v_P, t)\}$. Let the capacity of all (s, u_P) be $2a$, the capacity of all (u_P, v_P) be $2c$, the capacity of all (v_P, t) be $2b$. Because

the max flow in this graph is at least $\frac{f'}{\lceil \log n \rceil^3}$ and a, b, c are multiple of $\frac{1}{n}$, we can find a flow setting that maximizing the flow, and the flow of each path is multiple of $\frac{1}{n}$. Let B^* be the set of paths that has non-zero flow in this setting, and f^* be the flow, we know that $f^* \geq \frac{f'}{\lceil \log n \rceil^3}$.

This rearrangement of flow also tells us that the degree from S_1 to S_2 through paths in B^* is at most $\frac{2a}{c}$. This is because: If $c > \frac{1}{n}$, then degree from S_1 to S_2 through paths in B is at most $\frac{2a}{c}$, otherwise $c = \frac{1}{n}$, the degree from S_1 to S_2 through paths in B^* is at most $\frac{2a}{\frac{1}{n}}$.

For each path $P \in B^*$, we consider the degree of $v_P \in S_2$ that goes into S_1 (i.e. $|\{u_{P'} \mid P' \in B^*, v_{P'} = v_P\}|$), then we can construct $\lceil \log n \rceil$ buckets $B'_1, \ldots, B'_{\lceil \log n \rceil}$ on the degree:

$$B'_i = \{P \mid P \in B^*, 2^i \leq |\{u_{P'} \mid P' \in B^*, v_{P'} = v_P\}| \leq 2^{i+1}\}$$

Because the total flow is at least f^*, then there must be a bucket $B' \subseteq B^*$ that has flow at least $\frac{f^*}{\lceil \log n \rceil} \geq \frac{f'}{\lceil \log n \rceil^4}$. And there exist $d \in \mathbb{N}$, for each $P \in B'$, the degree of $v_P \in S_2$ that goes into S_1 is $|\{u_{P'} \mid P' \in B^*, v_{P'} = v_P\}| = |\{u_{P'} \mid P' \in B', v_{P'} = v_P\}| \in [d, 2d]$.

Because for all $v_P \in S_2$ with $P \in B'$, the degree goes into S_1 before bucketing is at least $\frac{b}{2c} \geq \frac{1}{2c\beta}$. And after bucketing, the average degree will decrease at most $\lceil \log n \rceil^4$ times. Therefore $d \geq \frac{1}{2c\beta \lceil \log n \rceil^4} \geq \frac{1}{2a\beta \lceil \log n \rceil^4}$.

For all $v_P \in S_2$ with $P \in B'$, the degree to S_1 is at most the number of nodes in S_1 which contains some flow in B, so $2d \leq \frac{\varepsilon\beta}{a}$ because $f \leq \beta$ and we only consider $f' \leq \varepsilon\beta$ flow.

Let $P \in B'$, we will show that if

$$r_{u_P} \leq \max\{\frac{273622a\lceil\log n\rceil^{13}}{\varepsilon^3}, \frac{64\lceil\log n\rceil^8}{\varepsilon^2 d}\} \text{ and } r_{v_P} \leq \frac{267c\beta\lceil\log n\rceil^5}{\varepsilon a} \quad (2.17)$$

then all edges of P are in $E_H \backslash F$. This is because if (2.17) holds, then $r_s r_{u_P} \leq 1 \leq a\alpha \leq \hat{x}_{(s,u_P)}\alpha$ and so $(s, u_P) \in E_H \backslash F$. Similarly, at the other end of the path we would have that $r_{v_P} r_t \leq 1 \leq b\alpha \leq \hat{x}_{(v_P,t)}\alpha$, so $(v_P, t) \in E_H \backslash F$. For the middle edge of the path, we would have that $r_{u_P} r_{v_P} \leq \max\{\frac{273622a\lceil\log n\rceil^{13}}{\varepsilon}, \frac{64\lceil\log n\rceil^8}{d}\} \cdot \frac{267c\beta\lceil\log n\rceil^5}{\varepsilon a}$.

If $\frac{64\lceil\log n\rceil^8}{d}$ is larger, we would have that $\frac{64\lceil\log n\rceil^8}{d} \leq c\alpha \leq \hat{x}_{(u_P,v_P)}\alpha$ because $d \geq \frac{1}{2c\beta\lceil\log n\rceil^4}$. So $(u_P, v_P) \in E_H$. Otherwise if $\frac{273622a\lceil\log n\rceil^{13}}{\varepsilon}$ is larger, we would have that $\frac{273622a\lceil\log n\rceil^{13}}{\varepsilon} \cdot \frac{267c\beta\lceil\log n\rceil^5}{\varepsilon a} = \frac{273622 \cdot 267c\beta\lceil\log n\rceil^{18}}{\varepsilon^3} \leq c\alpha \leq \hat{x}_{(u_P,v_P)}\alpha$, so $(u_P, v_P) \in E_H \backslash F$.

We want to use lemma 2.7.10 to show that with probability at least $1 - \frac{2}{n^{f+4}}$, we have enough many nodes in S_2 which connects to some nodes in S_1 with small enough r_{u_P}: $|\{v_P \mid P \in B', r_{u_P} \leq \max\{\frac{273622a\lceil\log n\rceil^{13}}{\varepsilon}, \frac{64\lceil\log n\rceil^8}{d}\}| \geq \frac{f'}{16cd\lceil\log n\rceil^4}$. Among all these v_P, the probability that there is no one has $r_{v_P} \leq \frac{267b\lceil\log n\rceil^5}{\varepsilon}$ is at most $(1 - \frac{267b\lceil\log n\rceil^5}{\varepsilon})^{\frac{f'}{16cd\lceil\log n\rceil^4}} \leq e^{-(f+4)\ln n}$ because $d \leq \frac{\varepsilon\beta}{2a}$. By union bound, we will know that with probability at least $1 - \frac{1}{n^{f+3}}$, there is a path in $E_H \backslash F$ from s to t with length 3.

Now we introduce how we use the lemma. Let $S_1' = \{u_P \mid P \in B'\}$, $S_2' = \{v_P \mid P \in B'\}$ and $E' = \{(u_P, v_P) \mid P \in B'\}$ be the set of nodes and edges remaining between S_1 and S_2 in this case. We know that (S_1', S_2', E') is a bipartite graph. Then $|S_1'| \leq \frac{f'}{a}$ when $a > \frac{1}{n}$ and $|S_1'| \leq n$ when $a = \frac{1}{n}$, so $|S_1'| \leq \frac{f'}{\varepsilon a}$ because $f' \geq \varepsilon$. Treat S_1' as the left part S_2' as the right part. Then

$|E'| \geq \frac{\frac{f'}{\lceil \log n \rceil^4}}{2c} = \frac{f'}{2c\lceil \log n \rceil^4}$. The maxium degree of the left part $\leq \frac{2a}{c}$. The average degree of the left part $\geq \frac{\frac{f'}{2c\lceil \log n \rceil^4}}{\frac{f'}{\varepsilon a}} \geq \frac{a}{2\varepsilon c \lceil \log n \rceil^4}$. The maxium degree of the right part $\leq 2d$, the average degree of the right part $\geq d$. So we can set $d_L = \frac{2a}{c}, r_L = \frac{4\lceil \log n \rceil^4}{\varepsilon}, d_R = 2d, r_R = 2$. We also set $p = \max\{\frac{273622a\lceil \log n \rceil^{13}}{\varepsilon^3}, \frac{64\lceil \log n \rceil^8}{\varepsilon^2 d}\}$.

If $p \geq 1$, then the lemma always works. Otherwise, $p < 1$, we know that $p \geq \frac{8r_L^2}{d_R}$ because $p \geq \frac{64\lceil \log n \rceil^8}{\varepsilon^2 d}$ and $\frac{Np}{256 d_L r_L^2} \geq (f+4)\ln n$ because $p \geq \frac{273622a\lceil \log n \rceil^{13}}{\varepsilon^3}$, the lemma also holds.

Lemma 2.7.10 gives a subset $S_2'' \subseteq S_2'$ such that $|S_2''| \geq \frac{|S_2'|}{2r_R} = \frac{|S_2'|}{4} \geq \frac{\frac{f'}{2c \cdot 2d \cdot \lceil \log n \rceil^4}}{4} = \frac{f'}{16cd\lceil \log n \rceil^6}$ with probability at least $1 - \frac{2}{n^{f+4}}$. This is what we want. □

Lemma 2.7.15. *For each $(s,t) \in E$ and $F \subseteq V$, if $\sum_{P \in \mathcal{P}^{(4)}_{s,t,F}} f_P^* \geq \frac{\varepsilon(f+198)}{6}$ and there is no scene which satisfies Lemma 2.7.4, then with probability at least $1 - \frac{1}{n^{f+3}}$, there is a path in $E_H \backslash F$ from s to t with length 4.*

Proof. We begin by randomly partitioning $V_{s,t} \backslash (\{s,t\} \cup F)$ to 3 parts S_1, S_2, S_3, Then the probability that each node is in partition S_j is $\frac{1}{3}$ for $j \in \{1,2,3\}$. Let $\mathcal{P}' \subseteq \mathcal{P}^{(4)}_{s,t,F}$ be the collection of paths of the form $s \to u \to v \to w \to t$ where $u \in S_1, v \in S_2, w \in S_3$. Clearly each path $P \in \mathcal{P}^{(4)}_{s,t,F}$ is in $\mathcal{P}'$ with probability $(\frac{1}{3})^3 = \frac{1}{27}$. So $\mathbb{E}[\sum_{P \in \mathcal{P}'} f_P^*] \geq \frac{\varepsilon(f+198)}{6} \cdot \frac{1}{27} = \frac{\varepsilon(f+198)}{162}$, and hence there is *some* partition S_1, S_2, S_3 where $\sum_{P \in \mathcal{P}'} f_P^* = f' \geq \frac{\varepsilon(f+198)}{162} \geq \varepsilon$. Fix this partition and collection of paths $\mathcal{P}'$.

For each $e \in E$, we define $\hat{f}_e = \sum_{P: P \in \mathcal{P}', e \in P} f_P^*$ be the total flow in $\mathcal{P}'$ that pass through e. For each $v \in V$, we define $\hat{l}_v = \sum_{P: P \in \mathcal{P}', v \in P} f_P^*$ be the total flow in $\mathcal{P}'$ that pass through v. For each $P \in \mathcal{P}'$, let u_P be the node of P in S_1, let v_P

be the node of P in S_2, and let w_P be the node of P in S_3 (so P is of the form $s \to u_P \to v_P \to w_P \to t$). We will bucket the paths in $\mathcal{P}'$ according to four values: $\hat{f}_{(s,u_P)}$, $\hat{f}_{(u_P,v_P)}$, $\hat{f}_{(v_P,w_P)}$ and $\hat{f}_{(w_P,t)}$. We construct $\lceil \log n \rceil^4$ buckects

$$B_{i,j,k,\ell} = \left\{ P \mid P \in \mathcal{P}', \hat{f}_{(s,u_P)} \in \begin{cases} (0, \frac{2}{n}], & \text{if } i = 0 \\ (\frac{2^i}{n}, \frac{2^{i+1}}{n}], & otherwise \end{cases}, \right.$$

$$\hat{f}_{(u_P,v_P)} \in \begin{cases} (0, \frac{2}{n}], & \text{if } j = 0 \\ (\frac{2^j}{n}, \frac{2^{j+1}}{n}], & otherwise \end{cases},$$

$$\hat{f}_{(v_P,w_P)} \in \begin{cases} (0, \frac{2}{n}], & \text{if } k = 0 \\ (\frac{2^k}{n}, \frac{2^{k+1}}{n}], & otherwise \end{cases},$$

$$\left. \hat{f}_{(w_P,t)} \in \begin{cases} (0, \frac{2}{n}], & \text{if } \ell = 0 \\ (\frac{2^\ell}{n}, \frac{2^{\ell+1}}{n}], & otherwise \end{cases} \right\}$$

There must be one bucket such that $\sum_{P \in B_{i,j,k,\ell}} f^*_P \geq \frac{f'}{\lceil \log n \rceil^4}$. To simplity our notation, we call this bucket B. Now we can find a, b, c, d be multiples of $\frac{1}{n}$ such that for each $P \in B$, we have

$$\hat{f}_{(s,U_P)} \in \begin{cases} (0, 2a], & \text{if } a = \frac{1}{n} \\ (a, 2a], & otherwise \end{cases},$$

$$\hat{f}_{(u_P,v_P)} \in \begin{cases} (0, 2c], & \text{if } c = \frac{1}{n} \\ (c, 2c], & otherwise \end{cases},$$

$$\hat{f}_{(v_P,w_P)} \in \begin{cases} (0, 2d], & \text{if } d = \frac{1}{n} \\ (d, 2d], & otherwise \end{cases},$$

$$\hat{f}_{(w_P,t)} \in \begin{cases} (0, 2b], & \text{if } b = \frac{1}{n} \\ (b, 2b], & otherwise \end{cases}$$

We can easily see that $c \leq a, d \leq b$, and the induced subgraph by B is a

$(s, t, a, c, d, b, 1, F, \sum_{P \in B} f_P^*)$-scene.

We further divide the paths P in B to two cases: whether $\hat{l}_{v_P}$ is at least $\frac{1}{\beta}$ or not. Then one of these two cases must have total flow at least $\frac{f'}{2\lceil \log n \rceil^4}$.

Case 1: $\sum_{P \in B: \hat{l}_{v_P} < \frac{1}{\beta}} f_P^* \geq \frac{f'}{2\lceil \log n \rceil^4}$

Consider the graph with node set $\{s, t\} \cup \bigcup_{P \in B: \hat{l}_{v_P} < \frac{1}{\beta}} \{v_P, u_P, w_P\}$ in this case and the edge set $\bigcup_{P \in B: \hat{l}_{v_P} < \frac{1}{\beta}} \{(s, u_P), (u_P, v_P), (v_P, w_P), (w_P, t)\}$. This is a $(s, t, a, c, d, b, \frac{1}{\beta}, F, \sum_{P \in B} f_P^*)$-scene.

Case 1.1: $a < \frac{1}{\alpha}, b < \frac{1}{\alpha}$

This case should not appear, otherwise because $c \leq a < \frac{1}{\alpha}, d \leq b < \frac{1}{\alpha}$, and a $(s, t, a, c, d, b, \frac{1}{\beta}, F, \sum_{P \in B} f_P^*)$-scene is also a $(s, t, a, c, d, b, \frac{1}{\beta}, \varnothing, \sum_{P \in B} f_P^*)$-scene, there exists a scene which satisfies Lemma 2.7.4.

Case 1.2: $a \geq \frac{1}{\alpha}, b < \frac{1}{\alpha}$

Consider the graph with node set $\{s, t\} \cup \bigcup_{P \in B: \hat{l}_{v_P} < \frac{1}{\beta}} \{v_P, u_P, w_P\}$ in this case and the edge set $\bigcup_{P \in B: \hat{l}_{v_P} < \frac{1}{\beta}} \{(s, u_P), (u_P, v_P), (v_P, w_P), (w_P, t)\}$. Let the capacity of all (s, u_P) be $2a$, the capacity of all (u_P, v_P) be $2c$, the capacity of all (u_P, w_P) be $2d$, the capacity of all (w_P, t) be $2b$. Because the max flow in this graph is at least $\frac{f'}{2\lceil \log n \rceil^4}$ and a, b, c, d are multiple of $\frac{1}{n}$, we can find a flow setting that maximizing the flow, and the flow of each path is multiple of $\frac{1}{n}$. Let B^* be the set of paths that has non-zero flow in this setting, and f^* be the flow, we know that $f^* \geq \frac{f'}{2\lceil \log n \rceil^4}$.

This rearrangement of flow also tells us that the degree from S_1 to S_2 through paths in B^* is at most $\frac{2a}{c}$. This is because: If $c > \frac{1}{n}$, then degree from S_1 to S_2 through paths in B is at most $\frac{2a}{c}$, otherwise $c = \frac{1}{n}$, the degree from S_1

to S_2 through paths in B^* is at most $\frac{2a}{\frac{1}{n}}$. Similarily, the degree from S_3 to S_2 through paths in B^* is at most $\frac{2b}{d}$, the degree from S_2 to S_3 through paths in B^* is at most $\frac{\hat{l}_{v_P}}{d}$.

Let $P \in B^*$ with $\hat{l}_{v_P} < \frac{1}{\beta}$, and note that if

$$r_{w_P} \leq b\alpha \text{ and } r_{v_P} \leq \frac{\alpha}{n} \tag{2.18}$$

then all edges of P are in $E_H \backslash F$. This is because if (2.18) holds, then $r_s r_{u_P} \leq 1 \leq a\alpha \leq \hat{x}_{(s,u_P)}\alpha$ and so $(s, u_P) \in E_H \backslash F$. Similarly, at the other end of the path we would have that $r_{w_P} r_t \leq r_{w_P} \leq b\alpha \leq \hat{x}_{(w_P,t)}\alpha$, so $(w_P, t) \in E_H \backslash F$. For the second edge of the path, we would have that $r_{u_P} r_{v_P} \leq r_{v_P} \leq \frac{\alpha}{n} \leq c\alpha \leq \hat{x}_{(u_P,v_P)}\alpha$, so $(u_P, v_P) \in E_H \backslash F$. Finally, for the third path edge we would have that $r_{v_P} r_{w_P} \leq r_{v_P} \leq \frac{\alpha}{n} \leq d\alpha \leq \hat{x}_{(v_P,w_P)}\alpha$, and so $(v_P, w_P) \in E_H \backslash F$.

Let $S_2' = \{v_P \mid P \in B^*\}$, $S_3' = \{w_P \mid P \in B^*\}$ and $E' = \{(v_P, w_P) \mid P \in B^*\}$ be the set of nodes and edges remaining in this case. We use Janson's inequality (Lemma 2.7.7) to bound the probability.

Let the overall set be $S_2' \cup S_3'$, $p_i = \begin{cases} \frac{\alpha}{n}, & \text{if } i \in S_2' \\ b\alpha, & \text{if } i \in S_3' \end{cases}$, and $\mathcal{S}$ be the set of endpoint sets of each edge in E'. Let X be the number of edges (v_P, w_P) such that $r_{w_P} \leq b\alpha$ and $r_{v_P} \leq \frac{\alpha}{n}$, then $\mathbb{E}[X] = b\alpha \cdot \frac{\alpha}{n} \cdot |E'| = \frac{b\alpha^2}{n}|E'|$.

We calculate Δ by each edge:

$$\begin{aligned}
\Delta &= \sum_{A\in\mathcal{S}} \Bigg(\sum_{B\in\mathcal{S}:A\cap B\neq\emptyset, A\cap B\in S_2'} \mathbf{Pr}[X_A = X_B = 1] \\
&\quad + \sum_{B\in\mathcal{S}:A\cap B\neq\emptyset, A\cap B\in S_3'} \mathbf{Pr}[X_A = X_B = 1] \Bigg) \\
&\leq |E'| \left(\sum_{B\in\mathcal{S}:A\cap B\neq\emptyset, A\cap B\in S_2'} \frac{\alpha}{n}\cdot(b\alpha)^2 + \sum_{B\in\mathcal{S}:A\cap B\neq\emptyset, A\cap B\in S_3'} b\alpha\cdot(\frac{\alpha}{n})^2 \right) \\
&\leq |E'| \left(\frac{\frac{1}{\beta}}{d}\cdot\frac{\alpha}{n}\cdot(b\alpha)^2 + \frac{2b}{d}\cdot b\alpha\cdot(\frac{\alpha}{n})^2 \right) \\
&= \frac{b\alpha^3(\frac{b}{\beta}+2\frac{b}{n})}{nd}|E'| \leq \frac{3b^2\alpha^3}{\beta nd}|E'|
\end{aligned}$$

Therefore

$$\mathbf{Pr}[X=0] \leq e^{-\frac{(\frac{b\alpha^2}{n}|E'|)^2}{2\frac{b\alpha^2}{n}|E'|+\frac{3b^2\alpha^3}{\beta nd}|E'|}} \leq e^{-\frac{(\frac{b\alpha^2}{n}|E'|)^2}{\frac{4b^2\alpha^3}{\beta nd}|E'|}} = e^{-\frac{d\alpha\beta|E'|}{4n}} \leq e^{-\frac{d\alpha\beta}{4n}\cdot\frac{f^*}{2d}} \leq \frac{1}{n^{f+3}}.$$

Thus with probability at least $1-\frac{1}{n^{f+3}}$, there is a path in $E_H\backslash F$ from s to t with length 4.

Case 1.3: $a < \frac{1}{\alpha}, b \geq \frac{1}{\alpha}$

This case is symmetric to case 1.2.

Case 1.4: $a \geq \frac{1}{\alpha}, b \geq \frac{1}{\alpha}$

Let $P \in B$ with $\hat{l}_{v_P} < \frac{1}{\beta}$, and note that if

$$r_{v_P} \leq \frac{\alpha}{n} \tag{2.19}$$

then all edges of P are in $E_H \backslash F$. This is because if (2.19) holds, then $r_s r_{u_P} \leq 1 \leq a\alpha \leq \hat{x}_{(s,u_P)}\alpha$ and so $(s, u_P) \in E_H \backslash F$. Similarly, at the other end of the path we would have that $r_{w_P} r_t \leq 1 \leq b\alpha \leq \hat{x}_{(w_P,t)}\alpha$, so $(w_P, t) \in E_H \backslash F$. For the second edge of the path, we would have that $r_{u_P} r_{v_P} \leq r_{v_P} \leq \frac{\alpha}{n} \leq c\alpha \leq \hat{x}_{(u_P,v_P)}\alpha$, so $(u_P, v_P) \in E_H \backslash F$. Finally, for the third path edge we would have that $r_{v_P} r_{w_P} \leq r_{v_P} \leq \frac{\alpha}{n} \leq d\alpha \leq \hat{x}_{(v_P,w_P)}\alpha$, and so $(v_P, w_P) \in E_H \backslash F$.

Because we only consider the nodes $v_P \in S_2$ which has $\hat{l}_{v_P} < \frac{1}{\beta}$ and the total flow is at least $\frac{f'}{2\lceil \log n \rceil^4}$, there are at least $\frac{\beta f'}{2\lceil \log n \rceil^4}$ nodes in S_2 which is in a path of B'. The probability that there is no $r_{v_P} \leq \frac{\alpha}{n}$ is at most $(1 - \frac{\alpha}{n})^{\frac{\beta f'}{2\lceil \log n \rceil^4}} \leq e^{-\frac{\alpha\beta f'}{2n\lceil \log n \rceil^4}} \leq \frac{1}{n^{f+3}}$.

Thus with probability at least $1 - \frac{1}{n^{f+3}}$, there is a path in $E_H \backslash F$ from s to t with length 4.

Case 2: $\sum_{P \in B: \hat{l}_{v_P} \geq \frac{1}{\beta}} f_P^* \geq \frac{f'}{2\lceil \log n \rceil^4}$

Consider the graph with node set $\{s, t\} \cup \bigcup_{P \in B: \hat{l}_{v_P} \geq \frac{1}{\beta}} \{v_P, u_P, w_P\}$ in this case and the edge set $\bigcup_{P \in B: \hat{l}_{v_P} \geq \frac{1}{\beta}} \{(s, u_P), (u_P, v_P), (v_P, w_P), (w_P, t)\}$. Let the capacity of all (s, u_P) be $2a$, the capacity of all (u_P, v_P) be $2c$, the capacity of all (u_P, w_P) be $2d$, the capacity of all (w_P, t) be $2b$. Because the max flow in this graph is at least $\frac{f'}{2\lceil \log n \rceil^4}$ and a, b, c, d are multiple of $\frac{1}{n}$, we can find a flow setting that maximizing the flow, and the flow of each path is multiple of $\frac{1}{n}$. Let B^* be the set of paths that has non-zero flow in this setting, and f^* be the flow, we know that $f^* \geq \frac{f'}{2\lceil \log n \rceil^4}$.

This rearrangement of flow also tells us that the degree from S_1 to S_2 through paths in B^* is at most $\frac{2a}{c}$. This is because: If $c > \frac{1}{n}$, then degree from S_1 to S_2 through paths in B is at most $\frac{2a}{c}$, otherwise $c = \frac{1}{n}$, the degree from S_1

to S_2 through paths in B^* is at most $\frac{2a}{\frac{1}{n}}$. Similarily, the degree from S_3 to S_2 through paths in B^* is at most $\frac{2b}{d}$.

For each path $P \in B^*$, we consider the degree of $v_P \in S_2$ on each side (i.e. $|\{u_{P'} \mid P' \in B^*, v_{P'} = v_P\}|$ and $|\{w_{P'} \mid P' \in B^*, v_{P'} = v_P\}|$), then we can construct $\lceil \log n \rceil^2$ buckets $B'_{1,1}, \ldots, B'_{\lceil \log n \rceil, \lceil \log n \rceil}$ on the degree:

$$B'_{i,j} = \{P \mid P \in B^*, 2^i \leq |\{u_{P'} \mid P' \in B^*, v_{P'} = v_P\}| \leq 2^{i+1},$$

$$2^j \leq |\{w_{P'} \mid P' \in B^*, v_{P'} = v_P\}| \leq 2^{j+1}\}$$

Because the total flow is at least f^*, then there must be a bucket $B' \subseteq B^*$ that has flow at least $\frac{f^*}{\lceil \log n \rceil^2} \geq \frac{f'}{2\lceil \log n \rceil^6}$. And there exist $d_1, d_3 \in \mathbb{N}$, for each $P \in B'$, the degree of $v_P \in S_2$ that goes into S_1 is $|\{u_{P'} \mid P' \in B^*, v_{P'} = v_P\}| = |\{u_{P'} \mid P' \in B', v_{P'} = v_P\}| \in [d_1, 2d_1]$, the degree of $v_P \in S_2$ that goes into S_3 is $|\{w_{P'} \mid P' \in B^*, v_{P'} = v_P\}| = |\{w_{P'} \mid P' \in B', v_{P'} = v_P\}| \in [d_3, 2d_3]$.

Because for all $v_P \in S_2$ with $P \in B'$, the degree goes into S_1 before bucketing is at least $\frac{\hat{l}_{v_P}}{2c} \geq \frac{1}{2c\beta}$. And after bucketing, the average degree will decrease at most $2\lceil \log n \rceil^6$ times. Therefore $d_1 \geq \frac{1}{4c\beta\lceil \log n \rceil^6} \geq \frac{1}{4a\beta\lceil \log n \rceil^6}$. Using the same method, we will get $d_3 \geq \frac{1}{4d\beta\lceil \log n \rceil^6} \geq \frac{1}{4b\beta\lceil \log n \rceil^6}$.

For all $v_P \in S_2$ with $P \in B'$, the degree to S_1 is at most the number of nodes in S_1 which contains some flow in B, so $2d_1 \leq \frac{\varepsilon\beta}{a}$ because $f \leq \beta$ and we only consider $f' \leq \varepsilon\beta$ flow. Using the same method, we will get $d_3 \leq \frac{\varepsilon\beta}{2b}$.

Without lose of generality, we assume $\frac{c}{a} \leq \frac{d}{b}$. Let $P \in B'$, and we will show

that if

$$r_{u_P} \leq \max\{\frac{10^8 a\lceil\log n\rceil^{19}}{\varepsilon^3}, \frac{256\lceil\log n\rceil^{12}}{\varepsilon^2 d_1}\} \tag{2.20}$$

$$r_{w_P} \leq \max\{\frac{10^{11} b\lceil\log n\rceil^{19}}{\varepsilon^3}, \frac{16384\lceil\log n\rceil^{12}}{\varepsilon^2 d_3}\} \tag{2.21}$$

$$r_{v_P} \leq \frac{28747 c\beta\lceil\log n\rceil^7}{a} \tag{2.22}$$

then all edges of P are in $E_H \backslash F$.

If (2.20) holds, then $r_s r_{u_P} \leq r_{u_P} \leq \max\{\frac{10^8 a\lceil\log n\rceil^{19}}{\varepsilon^3}, \frac{256\lceil\log n\rceil^{12}}{\varepsilon^2 d_1}\} \leq a\alpha \leq \hat{x}_{(s,u_P)}\alpha$ because $d_1 \geq \frac{1}{4a\beta\lceil\log n\rceil^6}$. So $(s, u_P) \in E_H \backslash F$.

Similarly, if (2.21) holds, we would have that

$$r_{w_P} r_t \leq r_{w_P} \leq \max\{\frac{10^{11} b\lceil\log n\rceil^{19}}{\varepsilon^3}, \frac{16384\lceil\log n\rceil^{12}}{\varepsilon^2 d_3}\} \leq b\alpha \leq \hat{x}_{(v_P,t)}\alpha,$$

because $d_3 \geq \frac{1}{4b\beta\lceil\log n\rceil^6}$. So $(v_P, t) \in E_H \backslash F$.

For the second edge of the path, if (2.20) and (2.22) holds, if $\frac{256\lceil\log n\rceil^{12}}{\varepsilon^2 d_1}$ is larger, we would have that $r_{u_P} r_{v_P} \leq r_{u_P} \leq \frac{256\lceil\log n\rceil^{12}}{\varepsilon^2 d_1} \leq c\alpha \leq \hat{x}_{(u_P,v_P)}\alpha$ because $d_1 \geq \frac{1}{4c\beta\lceil\log n\rceil^6}$. So $(u_P, v_P) \in E_H \backslash F$. Otherwise if $\frac{10^8 a\lceil\log n\rceil^{19}}{\varepsilon^3}$ is larger, we would have that $r_{u_P} r_{v_P} \leq \frac{10^8 a\lceil\log n\rceil^{19}}{\varepsilon^3} \cdot \frac{28747 c\beta\lceil\log n\rceil^7}{a} = \frac{10^8 \cdot 28747 c\beta\lceil\log n\rceil^{26}}{\varepsilon^3} \leq c\alpha \leq \hat{x}_{(u_P,v_P)}\alpha$, so $(u_P, v_P) \in E_H \backslash F$.

Finally, for the third edge of the path, if (2.21) and (2.22) holds, if $\frac{16384\lceil\log n\rceil^{12}}{\varepsilon^2 d_3}$ is larger, we would have that $r_{u_P} r_{v_P} \leq r_{u_P} \leq \frac{16384\lceil\log n\rceil^{12}}{\varepsilon^2 d_3} \leq d\alpha \leq \hat{x}_{(u_P,v_P)}\alpha$ because $d_3 \geq \frac{1}{4d\beta\lceil\log n\rceil^6}$. So $(u_P, v_P) \in E_H \backslash F$. Otherwise if $\frac{10^{11} b\lceil\log n\rceil^{19}}{\varepsilon^3}$ is larger, we would have that $r_{w_P} r_{v_P} \leq \frac{10^{11} b\lceil\log n\rceil^{19}}{\varepsilon^3} \cdot \frac{28747 c\beta\lceil\log n\rceil^7}{a} = \frac{10^{11} \cdot 28747 bc\beta\lceil\log n\rceil^{26}}{a\varepsilon} \leq \frac{10^{11} \cdot 28747 d\beta\lceil\log n\rceil^{26}}{\varepsilon^3} \leq d\alpha \leq \hat{x}_{(w_P,v_P)}\alpha$, so $(w_P, v_P) \in E_H \backslash F$.

The idea of proving this case is like this:

We want to use lemma 2.7.10 to first show that with probability at least $1 - \frac{2}{n^{f+4}}$, we have enough many nodes in S_2 which connects to some nodes in S_1 with small enough r_{u_P}: $|\{v_P \mid P \in B', r_{u_P} \leq \max\{\frac{10^8 a\lceil\log n\rceil^{19}}{\varepsilon}, \frac{256\lceil\log n\rceil^{12}}{d_1}\}\}| \geq \frac{f'}{32cd_1\lceil\log n\rceil^6}$. Then, we want to use lemma 2.7.10 again to show that with probability at least $1 - \frac{2}{n^{f+4}}$, we have enough many nodes in S_2 which connects to some nodes in S_1 with small enough r_{u_P}, and also connects to some nodes in S_3 with small enough r_{w_P}: $|\{v_P \mid P \in B', r_{u_P} \leq \max\{\frac{10^8 a\lceil\log n\rceil^{19}}{\varepsilon}, \frac{256\lceil\log n\rceil^{12}}{d_1}\}, r_{w_P} \leq \max\{\frac{10^{11} b\lceil\log n\rceil^{19}}{\varepsilon}, \frac{16384\lceil\log n\rceil^{12}}{d_3}\}\}| \geq \frac{f'}{128cd_1\lceil\log n\rceil^6}$. Among all these v_P, if it is the case that $\frac{28747c\beta\lceil\log n\rceil^7}{a} \geq 1$, there must be one $v_P \leq \frac{28747c\beta\lceil\log n\rceil^7}{a}$. Otherwise, if $\frac{28747c\beta\lceil\log n\rceil^7}{a} < 1$, the probability that there is no one has $r_{v_P} \leq \frac{28747c\beta\lceil\log n\rceil^7}{a}$ is at most $(1 - \frac{28747c\beta\lceil\log n\rceil^7}{a})^{\frac{f'}{128cd_1\lceil\log n\rceil^6}} \leq e^{-(f+4)\ln n}$ because $d_1 \leq \frac{\varepsilon\beta}{2a}$. By union bound, we will know that with probability at least $1 - \frac{1}{n^{f+3}}$, there is a path in $E_H \backslash F$ from s to t with length 4.

Now we introduce how we use the lemma.

For the first time, let $S_1' = \{u_P \mid P \in B'\}$, $S_2' = \{v_P \mid P \in B'\}$ and $E' = \{(u_P, v_P) \mid P \in B'\}$ be the set of nodes and edges remaining between S_1 and S_2 in this case. We know that (S_1', S_2', E') is a bipartite graph. Then $|S_1'| \leq \frac{f'}{a}$ when $a > \frac{1}{n}$ and $|S_1'| \leq n$ when $a = \frac{1}{n}$, so $|S_1'| \leq \frac{f'}{\varepsilon a}$ because $f' \geq \varepsilon$. Treat S_1' as the left part and S_2' as the right part. Then $|E'| \geq \frac{\frac{f'}{2\lceil\log n\rceil^6}}{2c} = \frac{f'}{4c\lceil\log n\rceil^6}$. The maxium degree of the left part $\leq \frac{2a}{c}$. The average degree of the left part $\geq \frac{\frac{f'}{4c\lceil\log n\rceil^6}}{\frac{f'}{\varepsilon a}} \geq \frac{\varepsilon a}{4c\lceil\log n\rceil^6}$. The maxium degree of the right part $\leq 2d_1$, the average degree of the right part $\geq d_1$. So we can set $d_L = \frac{2a}{c}, r_L = \frac{8\lceil\log n\rceil^6}{\varepsilon}, d_R = 2d_1, r_R = 2$. We also set $p = \max\{\frac{10^8 a\lceil\log n\rceil^{19}}{\varepsilon^3}, \frac{256\lceil\log n\rceil^{12}}{\varepsilon^2 d_3}\}$.

If $p \geq 1$, then the lemma always works. Otherwise, $p < 1$, we know that $p \geq \frac{8r_L^2}{d_R}$ because $p \geq \frac{256\lceil \log n \rceil^{12}}{d_1}$ and $\frac{Np}{256 d_L r_L^2} \geq (f+4)\ln n$ because $p \geq \frac{10^8 a \lceil \log n \rceil^{19}}{\varepsilon}$, the lemma also holds.

Lemma 2.7.10 gives a subset $S_2'' \subseteq S_2'$ such that $|S_2''| \geq \frac{|S_2'|}{2r_R} = \frac{|S_2'|}{4}$ with probability at least $1 - \frac{2}{n^{f+4}}$.

Suppose lemma 2.7.10 succeeded in the first time. For the second time, let $S_3'' = \{w_P \mid P \in B', v_P \in S_2''\}$, S_2'' and $E'' = \{(v_P, w_P) \mid P \in B', v_P \in S_2''\}$ be the set of nodes and edges remaining between S_3 and S_2 in this case. We know that (S_3'', S_2'', E'') is a bipartite graph. Then $|S_3''| \leq \frac{f'}{b}$ when $b > \frac{1}{n}$ and $|S_3''| \leq n$ when $b = \frac{1}{n}$, so $|S_3''| \leq \frac{f'}{\varepsilon b}$ because $f' \geq \varepsilon$. Treat S_3' as the left part and S_2' as the right part. Because we lose at most $\frac{3}{4}$ fraction of the nodes in S_2', and the degree of each node in S_2' is bounded by factor 2, therefore the edges remaining is at least $\frac{\frac{1}{4}\cdot 1}{2} = \frac{1}{8}$. Then $|E''| \geq \frac{\frac{f'}{2\lceil \log n \rceil^6}}{2d} \cdot \frac{1}{8} = \frac{f'}{32d\lceil \log n \rceil^6}$. The maxium degree of the left part $\leq \frac{2b}{d}$. The average degree of the left part $\geq \frac{\frac{f'}{32d\lceil \log n \rceil^6}}{\frac{f'}{\varepsilon b}} \geq \frac{\varepsilon b}{32d\lceil \log n \rceil^6}$. The maxium degree of the right part $\leq 2d_3$, the average degree of the right part $\geq d_3$. So we can set $d_L = \frac{2b}{d}, r_L = \frac{64\lceil \log n \rceil^6}{\varepsilon}, d_R = 2d_3, r_R = 2$. We also set $p = \max\{\frac{10^{11} b \lceil \log n \rceil^{19}}{\varepsilon^3}, \frac{16384\lceil \log n \rceil^{12}}{\varepsilon^2 d_3}\}$.

If $p \geq 1$, then the lemma always works. Otherwise, $p < 1$, we know that $p \geq \frac{8r_L^2}{d_R}$ because $p \geq \frac{16384\lceil \log n \rceil^{12}}{d_3}$ and $\frac{Np}{256 d_L r_L^2} \geq (f+4)\ln n$ because $p \geq \frac{10^{11} b \lceil \log n \rceil^{19}}{\varepsilon}$, the lemma also holds.

Lemma 2.7.10 gives a subset $S_2''' \subseteq S_2''$ such that $|S_2'''| \geq \frac{|S_2''|}{2r_R} = \frac{|S_2''|}{4} \geq \frac{|S_2'|}{16} \geq \frac{\frac{f'}{2c\cdot 2d_1 \cdot \lceil \log n \rceil^6}}{16} = \frac{f'}{64 c d_1 \lceil \log n \rceil^6}$ with probability at least $1 - \frac{2}{n^{f+4}}$. This is what we want. □

References

Chechik, Shiri, Michael Langberg, David Peleg, and Liam Roditty (2010). "Fault Tolerant Spanners for General Graphs". In: *SIAM J. Comput.* 39.7, pp. 3403–3423. DOI: 10.1137/090758039. URL: http://dx.doi.org/10.1137/090758039.

Althöfer, Ingo, Gautam Das, David Dobkin, Deborah Joseph, and José Soares (1993). "On sparse spanners of weighted graphs". In: *Discrete Comput. Geom.* 9.1, pp. 81–100. ISSN: 0179-5376. DOI: http://dx.doi.org/10.1007/BF02189308.

Berman, Piotr, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev (2013). "Approximation algorithms for spanner problems and Directed Steiner Forest". In: *Inf. Comput.* 222, pp. 93–107. DOI: 10.1016/j.ic.2012.10.007. URL: http://dx.doi.org/10.1016/j.ic.2012.10.007.

Dinitz, Michael and Robert Krauthgamer (2011a). "Directed spanners via flow-based linear programs". In: *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pp. 323–332. DOI: 10.1145/1993636.1993680. URL: http://doi.acm.org/10.1145/1993636.1993680.

Dinitz, Michael, Guy Kortsarz, and Ran Raz (2012). "Label Cover Instances with Large Girth and the Hardness of Approximating Basic k-Spanner". In: *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP)*. DOI: 10.1007/978-3-642-31594-7_25. URL: http://dx.doi.org/10.1007/978-3-642-31594-7_25.

Elkin, Michael and David Peleg (2000). "The Hardness of Approximating Spanner Problems". In: *STACS*, pp. 370–381.

Kortsarz, Guy (2001). "On the Hardness of Approximating Spanners". In: *Algorithmica* 30.3, pp. 432–450.

Dinitz, Michael and Robert Krauthgamer (2011b). "Fault-tolerant spanners: better and simpler". In: *Proceedings of the 30th Annual ACM Symposium*

on Principles of Distributed Computing (PODC). DOI: 10.1145/1993806.1993830. URL: http://doi.acm.org/10.1145/1993806.1993830.

Feldman, Moran, Guy Kortsarz, and Zeev Nutov (2012). "Improved approximation algorithms for Directed Steiner Forest". In: *Journal of Computer and System Sciences* 78.1, pp. 279 –292.

Chlamtác, Eden and Michael Dinitz (2014). "Lowest Degree k-Spanner: Approximation and Hardness". In: *Proceedings of APPROX/RANDOM*. Vol. 28. LIPIcs, pp. 80–95.

Chlamtác, E., M. Dinitz, and R. Krauthgamer (2012). "Everywhere-Sparse Spanners via Dense Subgraphs". In: *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 758–767.

Erdős, Paul (1964). "Extremal problems in graph theory". In: *Theory Graphs Appl., Proc. Symp. Smolenice 1963*, pp. 29–36.

Bondy, J.A and M Simonovits (1974). "Cycles of even length in graphs". In: *Journal of Combinatorial Theory, Series B* 16.2, pp. 97 –105. ISSN: 0095-8956. DOI: http://dx.doi.org/10.1016/0095-8956(74)90052-5. URL: http://www.sciencedirect.com/science/article/pii/0095895674900525.

Lazebnik, F., V. A. Ustimenko, and A. J. Woldar (1995). "A new series of dense graphs of high girth". In: *Bull. Amer. Math. Soc. (N.S.)* 32.1, pp. 73–79. ISSN: 0273-0979. DOI: 10.1090/S0273-0979-1995-00569-0. URL: http://dx.doi.org/10.1090/S0273-0979-1995-00569-0.

Thorup, Mikkel and Uri Zwick (2005). "Approximate Distance Oracles". In: *J. ACM* 52.1, pp. 1–24. ISSN: 0004-5411. DOI: 10.1145/1044731.1044732. URL: http://doi.acm.org/10.1145/1044731.1044732.

Lovasz, L., V. Neumann-Lara, and M. Plummer (1978). "Mengerian theorems for paths of bounded length". English. In: *Periodica Mathematica Hungarica* 9.4, pp. 269–276. ISSN: 0031-5303. DOI: 10.1007/BF02019432. URL: http://dx.doi.org/10.1007/BF02019432.

Chapter 3

Distance Oracles

3.1 Definitions and Preliminaries

We begin with some basic definitions, including formal definitions of the problems that we will be working on.

Definition 3.1.1. An approximate distance oracle with (m, a)-stretch, size s, preprocessing time g, and query time h is a pair of algorithms, *preprocess* and *query*, with the following properties.

- $preprocess(V, d, m, a, r)$ is a randomized preprocessing algorithm which takes as input a metric space (V, d), stretch bound (m, a), and random string r and outputs a data structure S where the expected output size is at most $\mathbb{E}_r[|S|] \leq s(|V|, m, a)$ and the expected preprocessing time is at most $g(|V|, m, a)$.
- *query* takes as input a data structure $S = preprocess(V, d, m, a, r)$ (the output of the preprocess algorithm) with two vertices $u, v \in V$, and outputs a value $d'(u, v) \in \mathbb{R}$ such that $d(u, v) \leq d'(u, v) \leq m \cdot d(u, v) + a$. The running time of *query* is at most $h(|V|, m, a)$.

We will frequently refer to these just as “distance oracles" rather than “approximate distance oracles" when the stretch bound is clear from context.

The query algorithm guarantees here are deterministic: the randomness only affects the size of the data structure. Note that one could easily define distance oracles so that either the correctness (with respect to the stretch bound) or the query running time (or both) hold only in expectation or with high probability, but as discussed in Chapter 1, essentially all existing distance oracles (and in particular the Thorup-Zwick distance oracle) have deterministic guarantees on the queries.

This naturally leads us to the following question: If we fix a particular distance oracle and metric space, can we find the *best* possible data structure? Here we will focus on the output size, not the preprocessing time (as long as the preprocessing time is polynomial). In other words, since the query algorithm work on *any* of the possible data structures which the preprocessing algorithm might output, can we actually find the smallest such data structure? This gives the following natural optimization problem.

Definition 3.1.2. Let $\mathcal{A} = (preprocess, query)$ be an approximate distance oracle. The $\mathcal{A}$-optimization problem takes as input a metric space (V, d) and a stretch bound (m, a), and the goal is to find a string r which minimizes $|preprocess(V, d, m, a, r)|$.

In this thesis we will focus on two distance oracles (Thorup-Zwick (Thorup and Zwick, 2005) and Pătraşcu-Roditty (Patrascu and Roditty, 2010)), so we now introduce these oracles.

3.1.1 Thorup-Zwick Distance Oracle

For every integer $k \geq 1$, Thorup and Zwick (Thorup and Zwick, 2005) provided an approximate distance oracle with $(2k-1, 0)$-stretch, size $O(n^{1+\frac{1}{k}})$, preprocessing time $O(kn^{2+\frac{1}{k}})$, and query time $O(k)$. We call this distance oracle TZ_k.

Their preprocessing algorithm first constructs a chain of subsets $\varnothing = A_k \subseteq A_{k-1} \subseteq \ldots \subseteq A_0 = V$ by repeated sampling. Each set A_i, where $i \in [k-1]$, is obtained by including each element of A_{i-1} independently with probability $n^{-\frac{1}{k}}$.

Let $R_{iu} = \{v \in A_{i-1} \mid d(u,v) < \min_{w \in A_i} d(u,w)\}$ for all $u \in V$ and $i \in [k]$ (where by convention $\min_{w \in \varnothing} d(u,w) = \infty$ for all $u \in V$ to handle the $i = k$ case). The output data structure is obtained by storing (in a 2-level hash table) the distance from each node u to each node in $\bigcup_{i=1}^{k} R_{iu}$.

The data structure also stores a little more information. Each vertex u remembers $k-1$ pivots: $\arg\min_{w \in A_i} d(u,w)$ for all $i \in [k-1]$, and the distance from u to these pivots. However, this is a negligible space cost. We will ignore the cost of storing the pivots when analyzing the size of the oracle.

Clearly the output data structure is determined once $A_1, \ldots, A_{k-1}$ are fixed. The size of the data structure is:

$$\begin{aligned} cost(A_1, \ldots, A_{k-1}, V, d) &= \sum_{u \in V} \sum_{i=1}^{k} |R_{iu}| \\ &= \sum_{u \in V} \sum_{i=1}^{k} \left| \{v \in A_{i-1} \mid d(u,v) < \min_{w \in A_i} d(u,w)\} \right|. \end{aligned}$$

We will refer to $\sum_{u \in V} |R_{iu}|$ as the cost in level i.

Let us look back on the definition of approximate distance oracle. The random string r is only used to generate A_i's, and the query algorithm will return a correct distance estimate no matter what the sets A_i are, but the size is determined by the sets. Therefore, the TZ_k-optimization problem is to find the subsets $\varnothing = A_k \subseteq A_{k-1} \subseteq \ldots \subseteq A_0 = V$ in order to minimize the total cost.

3.1.2 Pătraşcu-Roditty Distance Oracle

Pătraşcu and Roditty (Patrascu and Roditty, 2010) provided an approximate distance oracle with $(2,1)$-stretch, size $O(n^{\frac{5}{3}})$, preprocessing time $O(n^2)$, and query time $O(1)$. We call this distance oracle PR. Note that PR works only for unweighted graph metrics.

Their preprocessing algorithm first construct a set $A \subseteq V$ via a complicated correlated sampling (informally, they sample a large set and a small set, and then define A to be everything in the large set and everything contained in a ball around the small set delimited by the large set). The data structure consists of a 2-level hash table for the distance from each node in A to each node in V, as well as a 2-level hash table storing the distance between each pair $\{u, v\} \subseteq V$ such that $d(u,v) < \min_{w \in A} d(u,w) + \min_{w \in A} d(v,w) - 1$

As with Thorup-Zwick, the output data structure is completely determined once A is fixed. Let

$$R = \left\{ \{u,v\} \subseteq V \mid d(u,v) < \min_{w \in A} d(u,w) + \min_{w \in A} d(v,w) - 1 \right\}.$$

Then the size of the data structure is

$$
\begin{aligned}
& cost(A, V, d) \\
& = n \cdot |A| + |R| \\
& = n \cdot |A| + \left| \left\{ \{u, v\} \subseteq V \mid d(u, v) < \min_{w \in A} d(u, w) + \min_{w \in A} d(v, w) - 1 \right\} \right|.
\end{aligned}
$$

As before, the random string r is only used to generate the set A, and any $A \subseteq V$ gives a data structure on which the query algorithm works. Therefore, the PR-optimization problem is to find the subset $A \subseteq V$ in order to minimize the total cost.

3.1.3 Distance Oracles With Outliers

In some cases, a small set of outlier vertices may make the size of the data structure blow up. Yet in some applications it is acceptable to ignore these outliers. This was the motivation behind a line of work on distance oracles with slack (Chan et al., 2009; Chan, Dinitz, and Gupta, 2006), in which the data structure could ignore the stretch bound on a small fraction of the distances.

In this thesis, we consider the case that we can refuse to answer distance queries for some outlier vertices. In other words, we can essentially remove an outlier set F out of V when computing the distance oracle. This gives us the problem of optimizing distance oracle with outliers, in which we not only need to find the random string to determine the output data structure, we also need to find the set of outliers to minimize the final cost. More formally, we have the following type of problem.

Definition 3.1.3. Given approximate distance oracle $\mathcal{A} = (preprocess, query)$, the $\mathcal{A}$-optimization problem with outliers takes as input a metric space (V, d), a stretch bound (m, a), and a bound on the number of outliers $f \in \mathbb{N}$. The goal is to find a string r as well as a set $F \subseteq V$ where $|F| \leq f$, in order to minimize $|preprocess(V \backslash F, d, m, a, r)|$.

We will provide both true approximation results and (α, β)-bicriteria results, in which we slightly violate the bound on the number of outliers. Formally, an (α, β)-approximation algorithm for the $\mathcal{A}$-optimization problem with outliers is on algorithm which on any input $((V, d), (m, a), f)$ returns a solution with cost at most $\alpha \cdot OPT$ that has at most $\beta \cdot f$ outliers (where OPT is the minimum cost of any solution with at most f outliers).

3.1.4 Results and Techniques

With these definitions in hand, we can now formally state our results.

In Section 3.2 we discuss the problem of optimizing the 3-stretch Thorup-Zwick distance oracle, i.e., the TZ_2-optimization problem. It is straightforward to obtain an $O(\log n)$-approximation by reducing to the non-metric facility location problem.

Theorem 3.1.4. *There is an $O(\log n)$-approximation algorithm for TZ_2-optimization problem.*

To prove a matching lower bound, we use a reduction from Label Cover to the TZ_2-optimization problem. We use a proof which is similar to the proof of the hardness of Set Cover in (Vazirani, 2013) (based on (Feige, 1998)).

However, we cannot use a reduction directly from Set Cover since we will need some extra properties of the starting instances, and thus are forced to start from Label Cover. We introduce a new notion of (m, l, δ)-set families and show that these can still be plugged into existing hardness results to get the extra structural properties that we need. This lets us prove the following theorem:

Theorem 3.1.5. *Unless* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log \log n)})$*, the* TZ_2*-optimization problem does not admit a polynomial-time* $o(\log n)$*-approximation.*

For larger stretch values, a natural approach is to realize that a simple LP relaxation suffices to give Theorem 3.1.4 in the stretch 3 case, and try to extend this basic LP to larger stretches. In Section 3.3, we show that this does not work for the more general TZ_k-optimization problem: the integrality gap jumps up to become a polynomial. The instance is very simple: it is just the metric space formed by shortest paths on the n-cycle. It turns out to be straightforward to calculate the optimal fractional LP cost, but proving that the optimal integral solution is large is surprisingly involved.

Theorem 3.1.6. *The basic LP relaxation for the* TZ_k*-optimization problem has an* $\Omega(n^{\frac{1}{k} - \frac{1}{2^{k-1}}})$ *integrality gap when* $k > 2$*.*

In Section 3.4 we discuss the problem of optimizing the Pătraşcu-Roditty distance oracle. The basic LP and a simple rounding algorithm gives us an $O(\log n)$-approximation algorithm.

Theorem 3.1.7. *There is an* $O(\log n)$*-approximation algorithm for PR-optimization problem.*

A reduction from set cover problem also gives us a matching lower bound.

Theorem 3.1.8. *Unless* **P** $=$ **NP**, *the PR-optimization problem does not admit a polynomial-time* $o(\log n)$*-approximation.*

In Section 3.5 we move to the outliers setting. For both TZ_2- and PR-optimization problems, a semidefinite programming relaxation and a simple rounding algorithm gives us an $(O(\frac{\log n}{\varepsilon}), 1+\varepsilon)$-approximation algorithm. Using an SDP relaxation seems to be necessary – the corresponding LP relaxation requires violating the number of outliers by a factor of 2 rather than a factor of $1+\varepsilon$. We can also get a true approximation on TZ_2-optimization problem with outliers if the number of outliers is low. These results form the following theorems.

Theorem 3.1.9. *There is an* $(O(\frac{\log n}{\varepsilon}), 1+\varepsilon)$*-approximation algorithm for the* TZ_2*-optimization problem with outliers.*

Theorem 3.1.10. *There is an* $O(\log n)$*-approximation algorithm for* TZ_2*-optimization problem with outliers if the number of outliers is at most* $\sqrt{n}$*.*

Theorem 3.1.11. *There is an* $(O(\frac{\log n}{\varepsilon}), 1+\varepsilon)$*-approximation algorithm for the PR-optimization problem with outliers.*

3.2 TZ_2-Optimization Problem

We first give an $O(\log n)$-approximation for TZ_2-optimization (Theorem 3.1.4), and follow this with a matching lower bound.

3.2.1 Upper Bound: Proof of Theorem 3.1.4

We will prove our upper bound by a reduction to the non-metric facility location problem.

Definition 3.2.1. In the *non-metric facility location* problem we are given a set F of facilities, a set D of clients, an opening cost function $f : F \to \mathbb{R}^+$, and a connection cost function $c : D \times F \to \mathbb{R}^+$. The goal is to find the set $S \subseteq F$ which minimizes $\sum_{i \in S} f(i) + \sum_{i \in D} \min_{j \in S} c(i, j)$ (i.e. the sum of the opening and connection costs).

Non-metric facility location is a classic problem, and much is known about it, including the following upper bound due to Hochbaum.

Theorem 3.2.2 (Hochbaum, 1982)**.** *There is a polynomial time algorithm which gives an $O(\log n)$-approximation to the non-metric facility location problem.*

Hochbaum's algorithm is a greedy algorithm, but it is also straightforward to design an algorithm with similar bounds using an LP relaxation. Since it is not necessary we do not present the relaxation here, but generalizations of the relaxation will prove important in the more general TZ_k setting (see Section 3.3).

We now show that the TZ_2-optimization problem is essentially a special

case of non-metric facility location problem. First, simple arithmetic manipulation of the cost function of the TZ_2-optimization problem gives the following:

$$
\begin{aligned}
&cost(A_1, V, d) \\
&= \sum_{u \in V} |R_{1u}| + \sum_{u \in V} |R_{2u}| \\
&= \sum_{u \in V} \left| \{v \in V \mid d(u,v) < \min_{w \in A_1} d(u,w)\} \right| + \sum_{u \in V} |\{v \in A_1 \mid d(u,v) < \infty| \\
&= \sum_{u \in V} \left| \{v \in V \mid d(u,v) < \min_{w \in A_1} d(u,w)\} \right| + n|A_1| \\
&= \sum_{w \in A_1} n + \sum_{u \in V} \min_{w \in A_1} |\{v \in V \mid d(u,v) < d(u,w)\}| .
\end{aligned}
$$

Given an instance (V, d) of the TZ_2-optimization problem, we create an instance of non-metric facility location by setting $F = D = V$, opening costs $f(v) = n$ for all $v \in V$, and connection costs $c(u, w) = |\{v \in V \mid d(u,v) < d(u,w)\}|$ for all $u, w \in V$. Then the cost function of the TZ_2-optimization problem is exactly the same as the cost function of non-metric facility location problem. Therefore TZ_2 is a special case of non-metric facility location, which together with Theorem 3.2.2 implies Theorem 3.1.4.

3.2.2 Lower Bound

3.2.2.1 Overview

Proving an $\Omega(\log n)$ hardness of approximation (Theorem 3.1.5) turns out to be surprisingly difficult. Technically we reduce directly to TZ_2-optimization from a version of the Label Cover problem that corresponds to applying

parallel repetition (Raz, 1998) to 3SAT-5, which is a standard starting point for hardness reductions. Informally, though, we are “really" reducing from Set Cover: given an instance of Set Cover, we show how to create an instance of TZ_2-optimization where the cost of the optimal solution is the same (up to a constant and a polynomial scaling factor). But in order for our reduction to work, we actually need more than just an arbitrary Set Cover instance: we need a version of Set Cover in which it is hard even to cover *most* of the elements, not just all of them.

So we have to also give a new reduction from Label Cover to Set Cover, showing that even this version of Set Cover is hard. It turns out that Feige's reduction (Feige, 1998), reinterpreted by Vazirani (Vazirani, 2013), essentially already gives us what we need. We just need to analyze it a bit more carefully. In particular, a key component of this reduction is what Vazirani called (m, l)-*set systems*, which can be thought of as nearly-unbiased sample spaces. We generalize this notion to (m, l, δ)-set systems, given in the following definition.

Definition 3.2.3. A set B (the universe) and a collection of subsets $C_1, \ldots, C_m$ of B form an *(m, l, δ)-set system* if any l sets in $\{C_1, \ldots, C_m, \overline{C_1}, \ldots, \overline{C_m}\}$ whose union contains at least $(1 - \delta)|B|$ elements must include both C_i and $\overline{C_i}$ for some i.

An (m, l)-set system is just a $(m, l, 0)$-set system. While not all (m, l)-set systems are (m, l, δ)-set systems for larger δ, the construction of (m, l)-set systems in (Vazirani, 2013) actually does generalize directly to larger values of δ. With this tool in hand, we follow through the rest of the reduction and it gives us the type of Set Cover instances which we need. Technically

our reduction skips this step by going directly from Label Cover to TZ_2-optimization, but generating these kinds of Set Cover instances is intuitively what the first part of the reduction is doing.

3.2.2.2 Label Cover Problem

For the lower bound, we start from a hard Label Cover instance, and following the steps in proving the hardness of approximating Set Cover problem. Since the definition of the Label Cover problem is somewhat complex, we break it into parts: first defining an instance, a labelling, and then defining the problem. Note that we are using a specific setting where the parameters in the graph are strongly related, so it is slightly different from the definition of classic/general Label Cover problem.

Definition 3.2.4. A label cover instance consists of $(G = (V_1, V_2, E), \Sigma, \Pi)$ where

- G is a bipartite graph between vertex sets V_1 and V_2 and an edge set E. Let $V' = V_1 \cup V_2$
- G is left and right regular. Denote by Δ_1 and Δ_2 the degrees of vertices in V_1 and V_2 respectively.
- For each edge e, there is a constraint Π_e which is a bijection function from Σ to itself. The set of all constraints in G are $\Pi = \{\Pi_e : \Sigma \to \Sigma \mid e \in E\}$

Definition 3.2.5. A *labelling* of the graph, is a mapping $\sigma : V' \to \Sigma$ which assigns a label for each vertex of G. A labelling σ is said to satisfy an edge $e = (v_1, v_2)$ if and only if $\Pi_e(\sigma(v_1)) = \sigma(v_2)$.

The following definition is the problem which will be the starting point of our reduction.

Definition 3.2.6. In the LabelCover$_{n,r,\varepsilon}$ problem, we are given an instance (G, Σ, Π) of Label Cover where $|V_1| = (5n)^r, |V_2| = (5n)^r, |\Sigma| = 7^r, \Delta_1 = 15^r, \Delta_2 = 15^r$, and one of the following is true:

- There exists a labelling σ such that it satisfies all the edges e in G (in which case we say that the input is a YES instance), or
- For any labelling σ of the vertices, no more than $\varepsilon^r|E|$ edges are satisfied by σ (in which case we say that the input is a NO instance).

The goal is to determine whether the input is a YES or a NO instance.

Label Cover forms the starting point of many hardness of approximation results. Its hardness is a now-classical application of the PCP theorem (Ben-Or et al., 1988) and Raz's parallel repetition lemma (Raz, 1998), which give the following theorem.

Theorem 3.2.7 (Raz, 1998)**.** *There exists a constant $\varepsilon \geq 0$ such that LabelCover$_{n,r,\varepsilon}$ is not in* **P** *unless* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(r)})$.

For example, there exists a constant $\varepsilon \geq 0$ such that LabelCover$_{n,3\log\log n,\varepsilon}$ is not in **P** unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log\log n)})$. Starting from here, we will fix $r = 3\log\log n$, and ε be the constant which makes LabelCover$_{n,3\log\log n,\varepsilon}$ hard.

3.2.2.3 (m, l, δ)-Set System

We also need a (m, l, δ)-set system (see Definition 3.2.3) to do the reduction. We can construct a (m, l, δ)-set system by using a (l, γ)-independent collection of length m strings.

Definition 3.2.8. Let B be a collection (may contains repetitions) of binary strings of length m. B is *(l, γ)-independent* if the following inequality holds for every $i_1, i_2, \ldots, i_l$ and $a \in \{0,1\}^l$:

$$\left| \Pr_{x \in B}[x_{i_1} = a_1 \wedge \ldots \wedge x_{i_l} = a_l] - 2^{-l} \right| \leq \gamma.$$

A corollary of lemma 1 and construction 3 in (Alon et al., 1992) provides a explict construction of (l, γ)-independent collection.

Corollary 3.2.9. *For any $l \leq m$, there is an explict construction of a $(l, (1 - 2^{-l}) \cdot 2^{-l-1})$-independent collection of length m strings with $|B| = 4^{l+1}m^2$ in $|B|^{O(1)}$ time.*

With the corollary in hand, we can construct a (m, l, δ)-set system with the parameters we want.

Lemma 3.2.10. *For any $l \leq m$, there is an explicit construction of a $(m, l, 2^{-l-1})$-set system with $|B| = 4^{l+1}m^2$ in $|B|^{O(1)}$ time.*

Proof. Let B be the collection of length m strings in Corollary 3.2.9. Define $C_i = \{x \in B \mid x_i = 1\}$ for all $i \in [m]$, we will show that $(B; C_1, \ldots, C_m)$ is a $(m, l, 2^{-l-1})$-set system.

Assume that there exist $D_{i_1}, D_{i_2}, \ldots, D_{i_l}$ such that

$$\left| \bigcup_{j=1}^{l} D_{i_j} \right| \geq (1 - 2^{-l-1})|B|,$$

where each D_{i_j} is either C_{i_j} or $\overline{C_{i_j}}$ (note that this implies that there are no j and k such that $D_{i_j} = \overline{D_{i_k}}$). Define

$$a_j = \begin{cases} 0, & \text{if } D_{i_j} = C_{i_j} \\ 1, & \text{if } D_{i_j} = \overline{C_{i_j}} \end{cases}.$$

Let $S = \{x \mid x \in B, x_{i_1} = a_1, x_{i_2} = a_2, \ldots, x_{i_l} = a_l\}$. Because B is a $(l, (1 - 2^{-l}) \cdot 2^{-l-1})$-independent collection, we have

$$\begin{aligned} |S| &= |\{x \mid x \in B, x_{i_1} = a_1, x_{i_2} = a_2, \ldots, x_{i_l} = a_l\}| \\ &> (2^{-l} - (1 - 2^{-l}) \cdot 2^{-l-1})|B| \\ &> 2^{-l-1}|B|. \end{aligned}$$

On the other hand, note by construction, for all $x \in S$ and $j \in [l]$, we have $x \notin D_{i_j}$, which implies that $\left|\bigcup_{j=1}^{l} D_{i_j}\right| \leq |B| - |S| < (1 - 2^{-l-1})|B|$: a contradiction. □

3.2.2.4 Reduction

We now show how to use the set systems from the previous section to give a reduction from Label Cover to TZ_2-optimization problem.

Let $(G = (V_1, V_2, E), \Sigma, \Pi)$ be a $\text{LabelCover}_{n,r,\varepsilon}$ instance with $r = 3 \log \log n$, and let $(B; C_1, \ldots, C_m)$ be a $(m, l, 2^{-l-1})$-set system with $m = |\Sigma| = 7^r, l =$

$r \log n$.

We first create a universe $\mathcal{U} = E \times B$, and a set of sets $\mathcal{S} = \{S_{v,x} \mid v \in V', x \in [m]\}$ (recall that $V' = V_1 \cup V_2$). Here

$$S_{v,x} = \bigcup_{e:v \in e, e \in E} \{e\} \times C_{\Pi_e(x)}, \text{ if } v \in V_1,$$

$$S_{v,x} = \bigcup_{e:v \in e, e \in E} \{e\} \times \overline{C_x}, \text{ if } v \in V_2.$$

We know that $|E| = (15n)^r$, $|B| = 4^{r \log n + 1} \cdot 7^{2r} = n^{\Theta(1) \cdot r}$ and $|\mathcal{S}| = m \cdot |V'| = 7^r \cdot 2 \cdot (5n)^r$, so $|\mathcal{U}| \gg |\mathcal{S}|$. Without lose of generality and for simplicity of our proof, we can assume $|\mathcal{U}|$ is dividable by $|\mathcal{S}|$, so that we can replicate $\mathcal{S}$ for $\frac{|\mathcal{U}|}{|\mathcal{S}|}$ times, and get a set of sets $\mathcal{S}' = \mathcal{S}^{(1)} \cup \ldots \cup \mathcal{S}^{\left(\frac{|\mathcal{U}|}{|\mathcal{S}|}\right)}$ which has the same size as $\mathcal{U}$.

It is also easy to see that each $u = ((v_1, v_2), b) \in \mathcal{U}$ appears in exactly m sets in $\mathcal{S}$ because for each $x \in [m]$, either $u \in S_{v_1,x}$ or $u \in S_{v_2,\Pi_{(v_1,v_2)}(x)}$. Therefore each $u \in \mathcal{U}$ appears in $\frac{m|\mathcal{U}|}{|\mathcal{S}|} = \frac{|\mathcal{U}|}{|V'|}$ sets in $\mathcal{S}'$.

The metric space is defined as $V = \mathcal{U} \cup \mathcal{S}'$ and the distance is defined as following:

$$d(u,v) = \begin{cases} 1.2, & \text{if } u \in \mathcal{S}', v \in \mathcal{S}' \\ 1.4, & \text{if } u \in v \text{ or } v \in u \\ 1.6, & \text{if } u \in \mathcal{U}, v \in \mathcal{U} \\ 1.8, & \text{otherwise} \end{cases}$$

This metric space (V, d) will form the instance of TZ_2-optimization which we analyze. It is easy to see that the reduction is polynomial because $|V|$ is polynomial of $|E|$.

3.2.2.5 Analysis: Proof of Theorem 3.1.5

Lemma 3.2.11. *If (G, Σ, Π) is a YES instance in the LabelCover$_{n,r,\varepsilon}$ problem. Then the reduction (V, d) to the TZ_2-optimization problem has a solution with cost $\leq (|V'| + 1) \cdot |V|$.*

Proof. Let $\sigma : V' \to [m]$ denote a labelling of G which satisfies all the edges in E. Let $A_1 = \{S_{v,\sigma(v)} \mid v \in V'\}$. We claim that A_1 is a solution with cost at most $(|V'| + 1) \cdot |V|$.

Note that in Section 3.2.1 we showed that the level 2 cost $\sum_{u \in V} |R_{2u}| = |V| \cdot |A_1| = |V'| \cdot |V|$, the only thing left is to show that the level 1 cost is $\sum_{u \in V} |R_{1u}| \leq |V|$. We will prove this by showing $R_{1u} \subseteq \{u\}$ for all $u \in V$.

For any $u \in \mathcal{S}'$, we have that $d(u, v) \geq 1.2$ for all $v \in V$ because the definition of d, and $\min_{w \in A_1} d(u, w) = 1.2$ because $A_1 \cap \mathcal{S}' \neq \varnothing$. Thus $R_{1u} = \{v \in V \mid d(u, v) < \min_{w \in A_1} d(u, w)\} \subseteq \{u\}$.

For any $u = ((v_1, v_2), b) \in \mathcal{U}$, we have that $d(u, v) \geq 1.4$ for all $v \in V$ because the definition of d. We also know that either $u \in S_{v_1,\sigma(v_1)}$ or $u \in S_{v_1,\Pi_{(v_1,v_2)}(\sigma(v_1))}$ by the definition of $\mathcal{S}$, and $\Pi_{(v_1,v_2)}(\sigma(v_1)) = \sigma(v_2)$ because edge (v_1, v_2) is satisfied by labelling σ. Therefore $u \in S_{v_1,\sigma(v_1)} \cup S_{v_2,\sigma(v_2)}$. From the fact that both $S_{v_1,\sigma(v_1)}$ and $S_{v_2,\sigma(v_2)}$ are in A_1, we have $\min_{w \in A_1} d(u, w) = 1.4$. Thus $R_{1u} = \{v \in V \mid d(u, v) < \min_{w \in A_1} d(u, w)\} \subseteq \{u\}$.

Therefore $R_{1u} \subseteq \{u\}$ for all $u \in V$, so that A_1 is a solution with cost at most $\leq (|V'| + 1) \cdot |V|$. □

Lemma 3.2.12. *If (G, Σ, Π) is a No instance in the LabelCover$_{n,r,\varepsilon}$ problem. Then the reduction (V, d) to the TZ_2-optimization problem has no solution with cost*

$< \frac{l}{8}|V'| \cdot |V|$.

Proof. We prove the lemma by showing that if the optimal solution of the reduction (V, d) to the TZ_2-optimization problem has cost $< \frac{l}{8}|V'| \cdot |V|$, then there exists a labelling σ such that it satisfies more than $\varepsilon^r|E|$ edges.

Assume the optimal solution $A_1 \subseteq V$ has $cost(A_1, V, d) < \frac{l}{8}|V'| \cdot |V|$, then $|A_1| < \frac{l}{8}|V'|$ because the level 2 cost is $\sum_{u \in V} |R_{2u}| = |V||A_1|$.

Let $L_v = \{x \in [m] \mid \exists j \in \left[\frac{|\mathcal{U}|}{|\mathcal{S}|}\right] \text{ s.t. } S^{(j)}_{v,x} \in A_1 \cap \mathcal{S}'\}$ for all $v \in V$, then $\sum_v L_v \leq |A_1 \cap \mathcal{S}'| \leq |A_1| < \frac{l}{8}|V'|$. Therefore at least $\frac{3}{4}|V'|$ vertices has $|L_v| < \frac{l}{2}$, because otherwise $\sum_v L_v \geq (1 - \frac{3}{4}) \cdot |V'| \cdot \frac{l}{2} \geq \frac{l}{8}|V'|$.

Let $E_1 = \{e = (v_1, v_2) \in E \mid |L_{v_1}| < \frac{l}{2}, |L_{v_2}| < \frac{l}{2}\}$. Then $|E_1| \geq \frac{|E|}{2}$ because $|V_1| = |V_2| = \frac{|V'|}{2}$ and G is regular.

On the other hand, we define a $u \in \mathcal{U}$ is "uncovered" if $\{v \in A_1 \cap \mathcal{S}' \mid u \in v\} = \varnothing$. Then for any uncovered $u \in \mathcal{U}$, we know that $\min_{w \in A_1} d(u, w) = 1.6$. Thus

$$\begin{aligned} R_{1u} =& \{v \in V \mid d(u, v) < \min_{w \in A_1} d(u, w)\} \\ \geq& \{v \in \mathcal{S}' \mid d(u, v) < 1.6\} \\ =& \{v \in \mathcal{S}' \mid u \in v\} = \frac{|\mathcal{U}|}{|V'|}. \end{aligned}$$

Therefore $|\{u \in \mathcal{U} \mid u \text{ is uncovered}\}| < \frac{\frac{l}{8}|V'| \cdot |V|}{\frac{|\mathcal{U}|}{|V'|}} < \frac{l}{4}|V'|^2$.

Let $E_2 = \{e \in E \mid |\{u = (e, b) \in \mathcal{U} \mid u \text{ is uncovered}\}| < \frac{l|V'|^2}{|E|}\}$. Then $|E_2| \geq \frac{3}{4}|E|$ because otherwise $|\{u \in \mathcal{U} \mid u \text{ is uncovered}\}| \geq (|E| - \frac{3}{4}|E|) \cdot \frac{l|V'|^2}{|E|}\} \geq \frac{l}{4}|V'|^2$.

Let $E' = E_1 \cap E_2$, we know that $|E'| \geq \frac{|E|}{4}$.

Now, we will show that if we uniformly random sample labels from L_v for each $v \in V'$, the expected number of the edges satisfied in E' is at least $\frac{|E|}{l^2}$.

For each edge $e = (v_1, v_2) \in E'$ where $v_1 \in V_1$ and $v_2 \in V_2$. Assume $L_{v_1} = \{a_1, \ldots, a_p\}$, $L_{v_2} = \{b_1, \ldots, b_q\}$. Note that for every $e \in E_2$ we have

$$\left|\{u = (e, b) \in \mathcal{U} \mid \exists v \in A_1 \cap \mathcal{S}', u \in v\}\right| \geq |B| - \frac{l|V'|^2}{|E|},$$

and for all $u = ((v_1, v_2), b) \in \mathcal{U}$, there exists $v \in A_1 \cap \mathcal{S}'$ where $u \in v$ iff $u \in S_{v_1,a_i}$ or $u \in S_{v_2,b_i}$. Thus we have

$$\left|(\{e\} \times B) \cap \left((\bigcup_{i=1}^{p} S_{v_1,a_i}) \cup (\bigcup_{j=1}^{q} S_{v_2,b_j})\right)\right| \geq |B| - \frac{l|V'|^2}{|E|},$$

which means

$$\begin{aligned}
\left|(\bigcup_{i=1}^{p} C_{\Pi_e(a_i)}) \cup (\bigcup_{j=1}^{q} \overline{C_{b_j}})\right| &\geq |B| - \frac{l|V'|^2}{|E|} \\
&= (1 - \frac{l|V'|^2}{|E||B|})|B| \\
&= (1 - (\frac{5}{147n})^r l)|B| \\
&> (1 - 2^{-l-1})|B|.
\end{aligned}$$

Thus by the definition of $(m, l, 2^{-l-1})$)-set system, we know that there exists i, j such that $\Pi_e(a_i) = b_j$. Therefore, e is satisfied with probability $\frac{1}{|L_{v_1}| \cdot |L_{v_2}|} \geq \frac{4}{l^2}$ because the labels are uniformly sampled. Thus the expected number of the edges satisfied in E' is at least $\frac{4}{l^2} \cdot \frac{|E|}{4} = \frac{|E|}{l^2}$, which means, there is a way to

label all the vertices in V' and satisfies at least $\frac{|E|}{l^2}$ edges.

Finally, because $r = 3 \log\log n$ and $l = r \log n$, we know that at most $\varepsilon^r \cdot |E| < \frac{|E|}{l^2}$ edges can be satisfied by any labelling, which is a contradiction. □

With these lemmas, we can prove our lower bound on the TZ_2-optimization problem.

By Lemma 3.2.11 and Lemma 3.2.12, we have a polynomial reduction from LabelCover$_{n,r,\varepsilon}$ problem to TZ_2-optimization problem, which maps a YES instance of LabelCover$_{n,r,\varepsilon}$ to a TZ_2-optimization instance with optimal cost at most $(|V'| + 1) \cdot |V|$, and maps a NO instance of LabelCover$_{n,r,\varepsilon}$ to a TZ_2-optimization instance with optimal cost at least $\frac{l}{8}|V'| \cdot |V|$. The gap is $\frac{\frac{l}{8}|V'|\cdot|V|}{(|V'|+1)\cdot|V|} = \Theta(\frac{l}{8}) = \Theta(\log|V|)$.

Combined with the hardness Theorem 3.2.7, we know that unless **NP** $\subseteq$ **DTIME**$(n^{O(\log\log n)})$, the TZ_2-optimization problem does not admit a polynomial-time $o(\log n)$-approximation.

3.3 TZ_k-Optimization Problem

We now move to the more general TZ_k-optimization problem. While we are not able to give nontrivial upper bounds for this problem, we can at least show that the basic LP relaxation (as discussed in Section 3.2.1) does not give polylogarithmic bounds in this more general setting.

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{k} \sum_{u,v \in V} y_{uv}^{(i)} & & \\
\text{s.t.} \quad & 0 = x_v^{(k)} \leq x_v^{(k-1)} \leq \ldots \leq x_v^{(1)} \leq x_v^{(0)} = 1 & & \forall v \in V \\
& y_{uv}^{(i)} \geq x_v^{(i-1)} - \sum_{w \in B_u(v)} x_w^{(i)} & & \forall u, v \in V, i \in [k] \\
& y_{uv}^{(i)} \geq 0 & & \forall u, v \in V, i \in [k]
\end{aligned}
\tag{3.1}
$$

Figure 3.1: LP relaxation for TZ_k-Optimization Problem (LP_{TZ_k})

3.3.1 The LP

Let $B_u(v) = \{w \in V \mid d(u,w) \leq d(u,v)\}$. For every $v \in V$ and $i \in [k]$, let $x_v^{(i)}$ be a variable which is supposed to be an indicator for whether $v \in A_i$. Similarly, for all $u, v \in V$ and $i \in [k]$, let $y_{uv}^{(i)}$ be a variable which is supposed to be an indicator for whether $v \in R_{iu}$. (Recall that $R_{iu} = \{v \in A_{i-1} \mid d(u,v) < \min_{w \in A_i} d(u,w)\}$) We can easily write an LP relaxation for this problem. See Figure 3.1.

We first prove that our LP relaxation is indeed valid, i.e., we prove the following claim.

Claim 3.3.1. *LP_{TZ_k} is a valid relaxation to the TZ_k-optimization problem.*

Proof. Let $A_1, \ldots, A_{k-1}$ be a valid solution to the TZ_k-optimization problem. Let $x_v^{(i)} = \mathbb{1}_{v \in A_i}$ and $y_{uv}^{(i)} = \mathbb{1}_{v \in R_{iu}}$ for all $i \in [k]$ and $u, v \in V$. We can see that the objective value $\sum_{i=1}^{k} \sum_{u,v \in V} y_{uv}^{(i)} = \sum_{i=1}^{k} \sum_{u \in V} |R_{iu}| = cost(A_1, \ldots, A_{k-1}, V, d)$, which is the cost function.

We can also see that the first constraint is satisfied by $x_v^{(i)}$ and $y_{uv}^{(i)}$ because $\varnothing = A_k \subseteq A_{k-1} \subseteq \ldots \subseteq A_0 = V$. The second constraint is satisfied because

if $v \in A_{i-1}$ and there is no vertex in $A_i \cap B_u(v)$, then $v \in R_{iu}$. The third constraint is trivially satisfied.

Therefore $x_v^{(i)}$ and $y_{uv}^{(i)}$ is a valid solution to LP_{TZ_k} which makes the LP objective value equal to the actuall cost function. Thus the claim is proved. □

When restricted to the special case of $k = 2$, it is not hard to see that this LP is essentially a special case of the basic LP relaxation for non-metric facility location, which can be used to prove the $O(\log n)$ bound of Theorem 3.1.4. But for larger values of k the behavior is different, and does not result in a polylogarithmic integrality gap.

3.3.2 Integrality Gap

The integrality gap instance is quite simple: the metric (V, d) induced by shortest-path distances in a cycle. Slightly more formally, we let $V = [n]$, and use the cycle distance $d(u, v) = \min\{|u - v|, n + \min\{u, v\} - \max\{u, v\}\}$.

3.3.2.1 Cost of The LP Solution

We first show that on this instance, LP_{TZ_k} has a solution with low cost.

Lemma 3.3.2. *LP_{TZ_k} has a solution with cost $O(n^{1+\frac{1}{2^{k-1}}})$ on instance (V, d).*

Proof. Consider the following setting of the LP variables: let $x_v^{(i)} = n^{-\frac{2^i-1}{2^{k-1}}}$ for all $v \in V$ and $i \in [k-1]$, and let $y_{uv}^{(i)} = \max\{0, x_v^{(i-1)} - \sum_{w \in B_u(v)} x_w^{(i)}\}$ for all $u, v \in V$ and $i \in [k]$.

We can see that $x_v^{(i)} = n^{-\frac{2^i-1}{2^{k-1}}} \geq n^{-\frac{2^{i+1}-1}{2^{k-1}}} = x_v^{(i+1)}$ which satisfies the first constraint of LP_{TZ_k}, $y_{uv}^{(i)} \geq x_v^{(i-1)} - \sum_{w \in B_u(v)} x_w^{(i)}$ which satisfies the second

constraint of LP_{TZ_k}, and $y_{uv}^{(i)} \geq 0$ which satisfies the third constraint of LP_{TZ_k}. Therefore $x_v^{(i)}, y_{uv}^{(i)}$ is a valid solution to LP_{TZ_k}.

The objective value of this solution is

$$\sum_{i=1}^{k} \sum_{u,v \in V} y_{uv}^{(i)} \tag{3.2}$$

$$= \sum_{i=1}^{k} \sum_{u,v \in V} \max\{0, x_v^{(i-1)} - \sum_{w \in B_u(v)} x_w^{(i)}\} \tag{3.3}$$

$$= \sum_{i=1}^{k-1} \sum_{u,v \in V} \max\{0, x_v^{(i-1)} - |B_u(v)| x_v^{(i)}\} + \sum_{u,v \in V} (x_v^{(k-1)} - 0) \tag{3.4}$$

$$= \sum_{i=1}^{k-1} \sum_{u,v \in V} \max\{0, n^{-\frac{2^{i-1}-1}{2^{k-1}}} - (2 \cdot d(u,v) + 1) \cdot n^{-\frac{2^i-1}{2^{k-1}}}\} + \sum_{u,v \in V} n^{-\frac{2^{k-1}-1}{2^{k-1}}} \tag{3.5}$$

$$= \sum_{i=1}^{k-1} \sum_{u \in V} \left(n^{-\frac{2^{i-1}-1}{2^{k-1}}} - n^{-\frac{2^i-1}{2^{k-1}}} + n^{-\frac{2^{i-1}-1}{2^{k-1}}} - 3 \cdot n^{-\frac{2^i-1}{2^{k-1}}} + \dots \right) + n^{1+\frac{1}{2^{k-1}}} \tag{3.6}$$

$$= \sum_{i=1}^{k-1} n \cdot \left(O(n^{-\frac{2^{i-1}-1}{2^{k-1}}}) \cdot O(n^{\frac{2^i-1}{2^{k-1}} - \frac{2^{i-1}-1}{2^{k-1}}}) \right) + n^{1+\frac{1}{2^{k-1}}} \tag{3.7}$$

$$= \sum_{i=1}^{k-1} O(n^{1+\frac{1}{2^{k-1}}}) + n^{1+\frac{1}{2^{k-1}}} \tag{3.8}$$

$$= O(n^{1+\frac{1}{2^{k-1}}}) \tag{3.9}$$

Here equation (3.4) holds because of all $x_v^{(i)}$ are equal and all $x_v^{(k)} = 0$. Equation (3.5) holds because of the definition of circle distance. Equation (3.6) is a unrolling, and equation (3.7) is a summation over arithmetic progression. The last equation holds because of k is a constant. □

3.3.2.2 Optimal Solution of The Instance

Next we will show that the optimal solution of this instance is large.

Lemma 3.3.3. *The optimal solution to the instance (V, d) has cost at least $\Omega(n^{1+\frac{1}{k}})$.*

We will prove this lemma using a stronger claim. The lemma holds by setting $a = 1, b = \lfloor \frac{n}{2} \rfloor$, and $l = k$ in this claim:

Claim 3.3.4. *For a segment $[a, b]$ of the cycle where $a, b \in [n]$, $b - a < \frac{n}{2}$, and all the vertices in $[a, b]$ are NOT in A_l, we have $\sum_{i=1}^{l} \sum_{u \in [a,b] \cap [n]} |R_{iu}| \geq \left(\frac{b-a+1}{4^l}\right)^{1+\frac{1}{l}}$ for each $l \in [k]$.*

Proof. We prove this by doing induction on l. The base case is $l = 1$. For each vertex $u \in [a, b]$, We know that

$$R_{1u} = \{v \in V \mid d(u, v) < \min_{w \in A_1} d(u, w)\}$$

$$\subseteq \{v \in [a, b] \mid |u - v| \leq \min\{u - a, b - u\}\},$$

so $|R_{1u}| \geq 2 \cdot \min\{u - a, b - u\}$ because all the vertices in $[a, b]$ are NOT in A_1. So

$$\sum_{u \in [a,b]} |R_{1u}| \geq 2 \cdot (1 + 2 + \ldots + \left\lfloor \frac{b-a+2}{2} \right\rfloor + \ldots + 2 + 1) \geq \left(\frac{b-a+1}{4}\right)^2.$$

Now we consider general case $l \geq 2$, and assume the claim is established on $l - 1$.

Assume there are m vertices $t_1, \ldots, t_m \in [a, \lceil \frac{a+b}{2} \rceil] \cap A_{l-1}$, and $[a, \lceil \frac{a+b}{2} \rceil]$ are splitted to small segments $[a_0, b_0], \ldots, [a_m, b_m]$ where all the vertices in $[a_i, b_i]$ are not in A_{l-1} (if a segment has no vertex inside, we let $b_i = a_i - 1$

without lose of generality). Then for each $i \in [m]$ and $u \in [a_i, b_i + 1]$, we have $t_1, \ldots, t_i \in R_{lu}$ because $R_{lu} = \{v \in A_{l-1} \mid d(u,v) < \min_{w \in A_l} d(u,w)\}$. Thus

$$\begin{aligned}\sum_{i=1}^{l} \sum_{u \in [a,b] \cap [n]} |R_{iu}| &\geq \sum_{i=0}^{m} \left(\sum_{j=1}^{l-1} \sum_{u \in [a_i, b_i] \cap [n]} |R_{ju}| + \sum_{u \in [a_i, b_i] \cap [n]} |R_{lu}| \right) \\ &\geq \sum_{i=0}^{m} \left(\left(\frac{b_i - a_i + 1}{4^{l-1}} \right)^{1 + \frac{1}{l-1}} + i \cdot (b_i - a_i + 2) \right).\end{aligned}$$

If $m > \frac{b-a+1}{4}$, $\sum_{i=0}^{m} i \cdot 1$ is already at least $\left(\frac{b-a+1}{4^l} \right)^{1+\frac{1}{l}}$.

If $m \leq \frac{b-a+1}{4}$, we have a stronger inequality which we will prove later:

Lemma 3.3.5. *If $\alpha \in [1,2]$ and $x_i \geq 0$ for all $i \in [m]$, then*

$$\sum_{i=0}^{m} (x_i^{\alpha} + 4i \cdot x_i) \geq \left(\sum_{i=0}^{m} x_i \right)^{2 - \frac{1}{\alpha}}$$

Using this inequality, by setting $x_i = \frac{b_i - a_i + 1}{4^{l-1}}$ and $\alpha = 1 + \frac{1}{l-1}$ we have

$$\begin{aligned}\sum_{i=1}^{l} \sum_{u \in [a,b] \cap [n]} |R_{iu}| &\geq \left(\sum_{i=0}^{m} \frac{b_i - a_i + 1}{4^{l-1}} \right)^{2 - \frac{1}{1 + \frac{1}{l-1}}} \\ &\geq \left(\frac{\lceil \frac{a+b}{2} \rceil - a + 1 - m}{4^{l-1}} \right)^{1 + \frac{1}{l}} \\ &\geq \left(\frac{\frac{b-a}{2} + 1 - \frac{b-a}{4}}{4^{l-1}} \right)^{1 + \frac{1}{l}} \\ &\geq \left(\frac{b - a + 1}{4^l} \right)^{1 + \frac{1}{l}}.\end{aligned}$$

□

3.3.2.2.1 Proof of Lemma 3.3.5

Let $M = (\sum_{i=1}^{m} x_i)^{\frac{\alpha-1}{\alpha}}$. We first split the problem to 2 cases, depending on whether $m \leq M$.

Case 1: $m \leq M$.

In this case, by Hölder's inequality, we have

$$(\sum_{i=0}^{m} x_i^{\alpha})^{\frac{1}{\alpha}} \cdot (\sum_{i=0}^{m} 1^{\frac{\alpha}{\alpha-1}})^{\frac{\alpha-1}{\alpha}} \geq (\sum_{i=0}^{m} x_i \cdot 1)$$

thus

$$\sum_{i=0}^{m} x_i^{\alpha} \geq \frac{(\sum_{i=0}^{m} x_i)^{\alpha}}{m^{\alpha-1}} \geq \frac{(\sum_{i=0}^{m} x_i)^{\alpha}}{M^{\alpha-1}} \geq (\sum_{i=0}^{m} x_i)^{\alpha-\frac{\alpha-1}{\alpha}\cdot(\alpha-1)} = (\sum_{i=0}^{m} x_i)^{2-\frac{1}{\alpha}}$$

Case 2: $m > M$.

Let's fix $T = \sum_{i=0}^{m} x_i$ and consider the $\mathbf{x}^*$ which minimizes the left side: $l(\mathbf{x}) = \sum_{i=0}^{m}(x_i^{\alpha} + 4i \cdot x_i)$.

Consider any two consecutive variables x_j and x_{j+1}, we claim that, in $\mathbf{x}^*$, for each $0 \leq j < m$, either $x_{j+1}^* = 0$, or $(x_j^*)^{\alpha-1} - (x_{j+1}^*)^{\alpha-1} = \frac{4}{\alpha}$.

This is because, if we replace the x_{j+1} in $l(\mathbf{x})$ by $T - \sum_{i\neq(j+1)} x_i$ and do

partial derivative with respect of x_j, we have

$$\frac{\partial}{\partial x_j}\left(\sum_{i\neq(j+1)}(x_i^\alpha+4i\cdot x_i)+\left(T-\sum_{i\neq(j+1)}x_i\right)^\alpha+4(j+1)\cdot\left(T-\sum_{i\neq(j+1)}x_i\right)\right)$$

$$=\frac{\partial}{\partial x_j}\left(x_j^\alpha+4j\times x_j+\left(T-\textstyle\sum_{i\neq(j+1)}x_i\right)^\alpha+4(j+1)\cdot\left(T-\textstyle\sum_{i\neq(j+1)}x_i\right)\right)$$

$$=\alpha\cdot x_j^{\alpha-1}+4j-\alpha\cdot\left(T-\textstyle\sum_{i\neq(j+1)}x_i\right)^{\alpha-1}-4(j+1)$$

$$=\alpha\cdot\left(x_j^{\alpha-1}-\left(T-\textstyle\sum_{i\neq(j+1)}x_i\right)^{\alpha-1}\right)-4$$

$$=\alpha(x_j^{\alpha-1}-x_{j+1}^{\alpha-1})-4.$$

If we fix x_i for all $i\in[0,m]\cap\mathbb{N}\backslash\{j,j+1\}$, this partial derivative monotonically increases as x_j increases. Thus when $l(\mathbf{x})$ is minimized, either the partial derivative equals 0, which means $(x_j^*)^{\alpha-1}-(x_{j+1}^*)^{\alpha-1}=\frac{4}{\alpha}$, or x_j hits the ceiling, which means $x_j^*=T-\sum_{i\neq j,(j+1)}x_i^*$, so $x_{j+1}^*=0$.

This result shows that, the number series $(x_0^*)^{\alpha-1},(x_1^*)^{\alpha-1},\ldots,(x_m^*)^{\alpha-1}$ is in decreasing order, where

$$(x_i^*)^{\alpha-1}=\begin{cases}(x_{i-1}^*)^{\alpha-1}-\frac{4}{\alpha}, & \text{if } (x_i^*)^{\alpha-1}>\frac{4}{\alpha}\\ 0, & \text{otherwise}\end{cases}$$

If the number of non-zero entries in $\mathbf{x}^*$ is at most M, then this comes back to the Case 1. Otherwise, there are more than M non-zero entries in $\mathbf{x}^*$, thus

$x_0, x_1, \ldots, x_M$ are all non-zero, and $(x_i^*)^{\alpha-1} \geq \frac{4}{\alpha} \cdot (M-i)$ for $i \leq M$. Therefore

$$\sum_{i=0}^{m} (x_i^*)^{\alpha} \geq \sum_{i=0}^{M} \left(\frac{4}{\alpha} \cdot (M-i) \right)^{\frac{\alpha}{\alpha-1}} \tag{3.10}$$

$$\geq \sum_{i=1}^{M} \left(\frac{4i}{\alpha} \right)^{\frac{\alpha}{\alpha-1}} \tag{3.11}$$

$$\geq \frac{(\sum_{i=1}^{M} \frac{4i}{\alpha})^{\frac{\alpha}{\alpha-1}}}{M^{\frac{1}{\alpha-1}}} \tag{3.12}$$

$$\geq \frac{(\frac{2M^2}{\alpha})^{\frac{\alpha}{\alpha-1}}}{M^{\frac{1}{\alpha-1}}} \tag{3.13}$$

$$\geq (\frac{2}{\alpha})^{\frac{\alpha}{\alpha-1}} \cdot (\sum_{i=0}^{m} x_i)^{\frac{\alpha-1}{\alpha} \cdot (\frac{2\alpha}{\alpha-1} - \frac{1}{\alpha-1})} \tag{3.14}$$

$$\geq (\sum_{i=0}^{m} x_i)^{2-\frac{1}{\alpha}} \tag{3.15}$$

Here inequality (3.15) holds because of $\alpha \in [1,2]$. Inequality (3.12) holds because of Hölder's inequality

$$(\sum_{i=1}^{m} 1^{\alpha})^{\frac{1}{\alpha}} \cdot (\sum_{i=1}^{m} y_i^{\frac{\alpha}{\alpha-1}})^{\frac{\alpha-1}{\alpha}} \geq \sum_{i=1}^{m} y_i \cdot 1$$

□

3.3.2.3 Proof of Theorem 3.1.6

Combine Lemma 3.3.2 and Lemma 3.3.3, there is an $\Omega(\frac{n^{1+\frac{1}{k}}}{n^{1+\frac{1}{2^k-1}}}) = \Omega(n^{\frac{1}{k}-\frac{1}{2^k-1}})$ integrality gap for the basic LP relaxation LP_{TZ_k}.

$$
\begin{array}{lll}
\min & \sum_{v \in V} n \cdot x_v + \sum_{\{u,v\} \subseteq V} y_{uv} & \\
\text{s.t.} & y_{uv} \geq 1 - \sum_{w \in B_u(r) \cup B_v(d(u,v)-r)} x_w & \forall u, v \in V, \forall r \in [0, d(u,v)] \\
& x_v \in [0,1] & \forall v \in V \\
& y_{uv} \geq 0 & \forall u, v \in V
\end{array}
\tag{3.16}
$$

Figure 3.2: LP relaxation for PR-Optimization Problem (LP_{PR})

3.4 PR-Optimization Problem

We now move from Thorup-Zwick distance oracles to Pătraşcu-Roditty distance oracles. We show that from an optimization perspective, they are similar to TZ_2 in that we can find matching bounds: an $O(\log n)$-approximation, and $\Omega(\log n)$-hardness.

3.4.1 Upper Bound: Proof of Theorem 3.1.7

In this section we prove Theorem 3.1.7 by using an LP and randomized rounding to give an $O(\log n)$-approximation to the PR-optimization problem.

Let $B_u(v) = \{w \in V \mid d(u,w) \leq d(u,v)\}$, and $B(u,r) = \{w \in V \mid d(u,w) \leq r\}$. We can see $B_u(v) = B(u, d(u,v))$. Now, let x_v be a variable which is supposed to be an indicator for whether $v \in A$, and let y_{uv} be a variable which is supposed to be an indicator for whether $\{u,v\} \in R$. (Recall that $R = \{\{u,v\} \subseteq V \mid d(u,v) < \min_{w \in A} d(u,w) + \min_{w \in A} d(v,w) - 1\}$). We can write the LP relaxation given in Figure 3.2.

At first blush it may not be obvious that the first type of constraint in this LP really captures the characterization of pairs in R. But it is actually not that

hard to see that this is a valid relaxation.

Claim 3.4.1. *LP_{PR} is a valid relaxation to the PR-optimization problem.*

Proof. Let A be a valid solution to the PR-optimization problem. Let $x_v = \mathbb{1}_{v \in A}$ and $y_{uv} = \mathbb{1}_{\{u,v\} \in R}$ for all $u, v \in V$. We can see that the objective value $\sum_{v \in V} n \cdot x_v + \sum_{\{u,v\} \subseteq V} y_{uv} = n \cdot |A| + |R| = cost(A, V, d)$, which is the cost function.

We can also see that the first constraint is satisfied by x_v and y_{uv} because if $y_{uv} = 0$, we have $d(u,v) \geq \min_{w \in A} d(u,w) + \min_{w \in A} d(v,w) - 1$, then for all $r \in [0, d(u,v)]$, there must be a vertex in $A \cap (B_u(r) \cup B_v(d(u,v) - r))$, which makes $0 \geq 1 - \sum_{w \in B_u(r) \cup B_v(d(u,v)-r)} x_w$ satisfied. The second and the third constraints are trivially satisfied.

Therefore x_v and y_{uv} is a valid solution to LP_{PR} which makes the LP objective value equal to the actuall cost function. Thus the claim is proved. □

Note that while the number of constraints appears to be exponential (recall that we assume integer weights, but not necessarily unit weights, and hence $d(u,v)$ is not necessarily polynomial in the input size), it is in fact possible to solve this LP in polynomial time. We can do this by noting that for each $u, v \in V$, only at most n different value of r actually yield *different* constraints, so we can simply write the constraints for those values.

Our algorithm is relatively straightforward. We first solve LP_{PR} and get an optimal fractional solution (x_v^*, y_{uv}^*). We then use independent randomized rounding, adding each $v \in V$ to A independently with probability $\min\{4 \ln n \cdot x_v^*, 1\}$.

Lemma 3.4.2. *If $y^*_{uv} \leq \frac{1}{2}$, then the probability that $\{u, v\} \in R$ is at most $\frac{1}{n}$.*

Proof. If $y^*_{uv} \leq \frac{1}{2}$, then the first constraint implies that $\sum_{w \in B_u(r) \cup B_v(d(u,v)-r)} x^*_w \geq \frac{1}{2}$ for all $r \in [0, d(u,v)]$. Therefore, the probability that $A \cap (B_u(r) \cup B_v(d(u,v) - r)) = \varnothing$ for a specific $r \in [0, d(u,v)]$ is at most

$$\prod_{w \in B_u(r) \cup B_v(d(u,v)-r)} (1 - \min\{4 \ln n \cdot x^*_w, 1\}) \leq e^{-\sum_{w \in B_u(r) \cup B_v(d(u,v)-r)} 4 \ln n \cdot x^*_w} \leq \frac{1}{n^2}$$

A union bound over all the different values of r we used in our LP implies that the probability that there exists an $r \in [0, d(u,v)]$ where $A \cap (B_u(r) \cup B_v(d(u,v) - r)) = \varnothing$ is at most $\frac{1}{n^2} \cdot n = \frac{1}{n}$. We claim that the existence of such an r is implied by $\{u, v\} \in R$, and hence the probability that $\{u, v\} \in R$ is at most $\frac{1}{n}$. To see this, suppose that $\{u, v\} \in R$, i.e. suppose that $d(u,v) < \min_{w \in A} d(u,w) + \min_{w \in A} d(v,w) - 1$. Then if we set $r = \min_{w \in A} d(u,w) - 1$, this implies that $\min_{w \in A} d(v,w) > d(u,v) - r$. But then this would imply that no element of A is in $B_u(r) \cup B_v(d(u,v) - r)$. □

Let $OPT_{LP_{PR}}$ denote the optimal cost of LP_{PR}. Then the above lemma implies that the expected cost of the rounding algorithm is at most

$$\begin{aligned}
\mathbf{E}[n|A| + |R|] &\leq \sum_{v \in V} n \cdot x^*_v \cdot 4 \ln n + 2 \cdot \sum_{u,v \in V} y^*_{uv} + n^2 \cdot \frac{1}{n} \\
&\leq O(\log n) \cdot OPT_{LP_{PR}} + n \\
&\leq O(\log n) \cdot OPT
\end{aligned}$$

(where we use the fact that $OPT \geq \Omega(n)$). This completes the proof of Theorem 3.1.7.

3.4.2 $\Omega(\log n)$-hardness: Proof of Theorem 3.1.8

It is well known that Set Cover problem is hard to approximate:

Theorem 3.4.3 (Raz and Safra, 1997)**.** *Unless* **P** $=$ **NP**, *there is no* $o(\log n)$*-approximation to the set cover problem.*

We now show a matching hardness bound for the PR-optimization problem by reducing from the Set Cover problem.

Consider a set cover instance $(\mathcal{U}, \mathcal{S})$ where $|\mathcal{U}| + |\mathcal{S}| = n$. For each $e \in \mathcal{U}$, we create a group of vertices G_e where $|G_e| = 3n$. For each $S \in \mathcal{S}$, we also create a group of vertices G_S where $|G_S| = 3n$.

Now we construct the following metric space: $V = (\bigcup_{e\in\mathcal{U}} G_e) \cup (\bigcup_{S\in\mathcal{S}} G_S)$ and

$$d(u,v) = \begin{cases} 1, & \text{if } u \in G_e, v \in G_e \\ 1, & \text{if } u \in G_S, v \in G_S \\ 1, & \text{if } u \in G_e, v \in G_S, e \in S \\ 2, & \text{otherwise.} \end{cases}$$

We now prove two lemmas about the reduction (completeness and soundness).

Lemma 3.4.4. *If there is a solution* $\mathcal{S}^*$ *to the set cover instance* $(\mathcal{U}, \mathcal{S})$ *where* $|\mathcal{S}^*| = t$*, then there is a set* A *where* $cost(A, V, d) \leq t|V|$*.*

Proof. For each $S \in S^*$, we add an arbitrary element from G_S to A. Then for every vertex in V, the closest vertex in A has distance at most 1 to it. Therefore

$$R = \left\{ \{u,v\} \subseteq V \mid d(u,v) < \min_{w\in A} d(u,w) + \min_{w\in A} d(v,w) - 1 \right\}$$

$$= \{\{u,v\} \subseteq V \mid d(u,v) < 1 + 1 - 1\} = \varnothing$$

Thus the total cost is at most $|V| \cdot |A| + |R| = t|V|$. □

Lemma 3.4.5. *If there is a set $A \subseteq V$ where $cost(A, V, d) \leq t|V|$, then there exists a solution $\mathcal{S}^*$ to the set cover instance $(\mathcal{U}, \mathcal{S})$ where $|\mathcal{S}^*| = t$.*

Proof. First, we say that a group $G = G_e$ or $G = G_S$ is "covered" if there exists a vertex $u \in G$, which $\min_{w \in A} d(u, w) = 1$. Then by the definition of d, it's easy to see that if a group G is covered, then for all vertices $u \in G$, we have $\min_{w \in A} d(u, w) = 1$. In addition, if a group G_e is covered, then either $G_e \cap A \neq \varnothing$, or there is a $S \in \mathcal{S}$, where $e \in S$ and $G_S \cap A \neq \varnothing$.

We can also see that, if a group G is not covered, then let

$$\begin{aligned} R_G &= \left\{ \{u, v\} \subseteq G \mid d(u, v) < \min_{w \in A} d(u, w) + \min_{w \in A} d(v, w) - 1 \right\} \\ &= \{\{u, v\} \subseteq G \mid d(u, v) < 2 + 2 - 1\} \\ &= \{\{u, v\} \subseteq G\} \end{aligned}$$

Thus $|R_G| \geq \frac{3n(3n-1)}{2} > 3n^2 > |V|$. Therefore if we add an arbitrary element from G to A, then $|R|$ decreases by at least $|R_G| \geq |V|$, and $|V| \cdot |A|$ increases by $|V|$, which makes $cost(A, V, d)$ only decrease. If we keep doing this operation, there will be a set A which makes sure that all the groups are covered, and $cost(A, V, d) \leq t|V|$. Now, for every vertex $u \in V$, we have $\min_{w \in A} d(u, w) = 1$, so $R = \varnothing$.

We can keep modifying A to the form we want. If there is a vertex $v \in G_e \cap A$, removing v and simultaneously adding a vertex in any $S \ni e$ to A does not increase the cost. This is because this operation keeps the fact that all

the groups are covered.

Finally, we have a set A where only contains vertices in $\bigcup_{S \in \mathcal{S}} G_S$ and $cost(A, V, d) \leq t|V|$. Let $\mathcal{S}^* = \{S \in \mathcal{S} \mid G_S \cap A \neq \varnothing\}$. Then $|\mathcal{S}^*| \leq t$ because $cost(A, V, d) = |A| \cdot |V| + |R| \geq |\mathcal{S}^*| \cdot |V|$, and $\mathcal{S}^*$ covers $\mathcal{U}$ because all the group G_e are covered. □

These lemmas, combined with Theorem 3.4.3, imply Theorem 3.1.8

3.5 Distance Oracles With Outliers

We now move to the more difficult outliers setting, where we can also optimize over a set of vertices to ignore. Recall that for an approximate distance oracle $\mathcal{A}$, our goal is now to find a set of vertices F (the outliers) where $|F| \leq f$ as well as a string r in order to minimize $|preprocess(V \setminus F, d, m, a, r)|$. In other words, we are going to try to solve the same problems as before, but where we can choose a set F to remove. We begin with TZ_2, and then move to PR.

3.5.1 TZ_2-Optimization Problem With Outliers

For this problem, it is easy to see that the cost function becomes:

$$\begin{aligned} cost(A, F, V, d) &= (n - f)|A| + \sum_{u \in V \setminus F} |R_{1u}| \\ &= (n - f)|A| + \sum_{u \in V \setminus F} \left| \{v \in V \setminus F \mid d(u, v) < \min_{w \in A} d(u, w)\} \right| \end{aligned}$$

A natural approach is to use an LP which is similar to LP_{TZ_k} to solve this problem (but for TZ_2), suitably adapted to handle outliers. Let x_v be a variable

$$
\begin{array}{lll}
\min & \displaystyle\sum_{v \in V} (n-f) \cdot x_v + \sum_{u,v \in V} y_{uv} & \\
\text{s.t.} & \displaystyle y_{uv} \geq 1 - z_u - z_v - \sum_{w \in B_u(v)} x_w & \forall u, v \in V \\
& \displaystyle\sum_{v \in V} z_v \leq f & \\
& x_v \in [0,1] & \forall v \in V \\
& y_{uv} \geq 0 & \forall u, v \in V \\
& z_v \in [0,1] & \forall v \in V
\end{array}
\tag{3.17}
$$

Figure 3.3: LP relaxation for TZ_2-Optimization Problem With Outliers (LP_{TZ_2O})

which is supposed to be an indicator for whether $v \in A$, let y_{uv} be a variable which is supposed to be an indicator for whether $v \in R_{1u}$, and let z_v be a variable which is supposed to be an indicator for whether $v \in F$. Then we can write the natural LP relaxation given in Figure 3.3.

Unfortunately, this LP can not give an (α, β)-approximation with $\beta = 2 - \epsilon$. To see this, consider the case that $f = \frac{n}{2}$. Then the optimal solution to LP_{TZ_2O} is 0, by setting all z_v to $\frac{1}{2}$, all x_v to 0, and all $y_{uv} = 0$. Thus any integral solution, to be competitive with this fractional solution, must treat *all* nodes as outliers, requiring β to be at least 2.

Fortunately we can give a stronger semidefinite programming relaxation, allowing for a better approximation. As in LP_{TZ_2O}, let $\vec{x}_v$ be a variable which is supposed to be an indicator for whether $v \in A$, let $\vec{y}_{uv}$ be a variable which is supposed to be an indicator for whether $v \in R_{1u}$, and let $\vec{z}_v$ be a variable which is supposed to be an indicator for whether $v \in F$. We can then write the SDP given in Figure 3.4.

$$
\begin{array}{lll}
\min & \sum_{v\in V}(n-f)\cdot\|\vec{x}_v\|^2+\sum_{u,v\in V}\|\vec{y}_{uv}\|^2 & \\
\text{s.t.} & \|\vec{y}_{uv}\|^2\geq 1-\vec{z}_u\cdot\vec{z}_v-\sum_{w\in B_u(v)}\|\vec{x}_w\|^2 & \forall u,v\in V \\
& \sum_{v\in V}\|\vec{z}_v\|^2\leq f & \\
& \|\vec{x}_v\|^2\leq 1 & \forall v\in V \\
& \|\vec{y}_{uv}\|^2\leq 1 & \forall u,v\in V \\
& \|\vec{z}_v\|^2\leq 1 & \forall v\in V
\end{array}
\tag{3.18}
$$

Figure 3.4: SDP relaxation for TZ_2-Optimization Problem With Outliers (SDP_{TZ_2O})

3.5.1.1 Bicriteria Approximation: Proof of Theorem 3.1.9

Our approximation algorithm first solves SDP_{TZ_2O} to get an optimal solution $(\vec{x}_v^*, \vec{y}_{uv}^*, \vec{z}_v^*)$. We then use independent randomized rounding to construct A, adding each $v \in V$ to A independently with probability $\min\{\frac{3\ln n}{\varepsilon}\cdot\|\vec{x}_v^*\|^2, 1\}$ where ε is a small constant. Finally, we use threshold rounding to construct F by adding each $v \in V$ to F if $\|\vec{z}_v^*\|^2 \geq \frac{1}{1+\varepsilon}$.

We want to show that this is an $(O(\log n), 1+\varepsilon)$-approximation. It is easy to see that $|F| \leq (1+\varepsilon)f$ because $\sum_{v\in V}\|\vec{z}_v^*\|^2 \leq f$. In order to prove Theorem 3.1.9 it only remains to prove that the expected cost is at most $O(\log n)\cdot OPT$.

Lemma 3.5.1. *If $\|\vec{y}_{uv}^*\|^2 \leq \frac{\varepsilon}{2}$, then the probability that $uv \in R_{1u}$ is at most $\frac{1}{n}$.*

Proof. If $\|\vec{z}_u^*\|^2 \geq \frac{1}{1+\varepsilon}$ or $\|\vec{z}_v^*\|^2 \geq \frac{1}{1+\varepsilon}$, then u or v is in F, so $v \notin R_{1u}$. Thus we only consider the case that $\|\vec{z}_u^*\|^2 < \frac{1}{1+\varepsilon}$ and $\|\vec{z}_v^*\|^2 < \frac{1}{1+\varepsilon}$, which means $\vec{z}_u^*\cdot\vec{z}_v^* < \frac{1}{1+\varepsilon}$. Since $\|\vec{y}_{uv}^*\|^2 \leq \frac{\varepsilon}{2}$, we have

$$\sum_{w\in B_u(v)}\|\vec{x}_w^*\|^2 \geq 1-\frac{\varepsilon}{2}-\frac{1}{1+\varepsilon} \geq \frac{\varepsilon}{3}.$$

Therefore, the probability that $d(u,v) < \min_{w \in A} d(u,w)$ is at most

$$\prod_{w \in B_u(v)} (1 - \min\{\frac{3 \ln n}{\varepsilon} \cdot \|\vec{x}_v^*\|^2, 1\}) \leq e^{-\sum_{w \in B_u(v)} \frac{3 \ln n}{\varepsilon} \cdot \|\vec{x}_v^*\|^2} \leq \frac{1}{n}$$

□

Therefore, let $OPT_{SDP_{TZ_2O}}$ denotes the optimal cost of SDP_{TZ_2O}, then the expected cost of the rounding algorithm is at most

$$\sum_{v \in V} (n - f) \cdot \frac{3 \ln n}{\varepsilon} \cdot \|\vec{x}_v^*\|^2 + \frac{2}{\varepsilon} \cdot \sum_{u,v \in V} \|\vec{y}_{uv}^*\|^2 + n^2 \cdot \frac{1}{n}$$

$$\leq O(\log n) \cdot SDP_{TZ_2O} + n$$

$$\leq O(\log n) \cdot OPT,$$

because $OPT \geq \Omega(n)$, which proves Theorem 3.1.9.

3.5.1.2 True approximation: Proof of Theorem 3.1.10

When $f \leq \sqrt{n}$ we can actually give a true $O(\log n)$-approximation (Theorem 3.1.10). The algorithm is almost the same; we just need to change the threshold rounding for outliers to instead pick the f vertices with largest $\|\vec{z}_v\|^2$ value.

When the number of outliers is low, in particular when $f \leq \sqrt{n}$, we can find an actual $O(\log n)$-approximation.

The SDP and rounding algorithm are the same, except we will choose f vertices with the highest $\|\vec{z}_v^*\|^2$ values as F, rather than a threshold rounding of $\frac{1}{1+\varepsilon}$.

Now there are two cases when $\|\vec{y}_{uv}^*\|^2 \leq \frac{\varepsilon}{2}$. One case is the same as before,

where $\sum_{w \in B_u(v)} \|\vec{x}_w^*\|^2 \geq \frac{\varepsilon}{3}$. In this case, the probability that $v \in R_{1u}$ is at most $\frac{1}{n}$. The other case is $\sum_{w \in B_u(v)} \|\vec{x}_w^*\|^2 < \frac{\varepsilon}{3}$, which means $\vec{z}_u^* \cdot \vec{z}_v^* \geq 1 - \frac{\varepsilon}{2} - \frac{\varepsilon}{3} = 1 - \frac{5}{6}\varepsilon$.

However, this case will not appear a lot. Whenever $\vec{z}_u^* \cdot \vec{z}_v^* \geq 1 - \frac{5}{6}\varepsilon$, both $\|\vec{z}_u^*\|$ and $\|\vec{z}_v^*\|$ should be at least $1 - \frac{5}{6}\varepsilon$, which means $\|\vec{z}_u^*\|^2$ and $\|\vec{z}_v^*\|^2$ is at least $\frac{1}{2}$. Because $\sum_{v \in V} \|\vec{z}_v^*\|^2 \leq f$, we know that there are at most $2f$ of $\|\vec{z}_v^*\|^2$ are at least $\frac{1}{2}$. Therefore the number of u, v pairs that $\|\vec{y}_{uv}^*\|^2 \leq \frac{\varepsilon}{2}$ and $\sum_{w \in B_u(v)} \|\vec{x}_w^*\|^2 < \frac{\varepsilon}{3}$ is at most $2f \cdot 2f = 4n$.

Therefore, let $OPT_{SDP_{TZ_2O}}$ denotes the optimal cost of SDP_{TZ_2O}, then the expected cost of the rounding algorithm is at most

$$\begin{aligned}
&\sum_{v \in V} (n - f) \cdot \frac{3 \ln n}{\varepsilon} \cdot \|\vec{x}_v^*\|^2 + \frac{2}{\varepsilon} \cdot \sum_{u,v \in V} \|\vec{y}_{uv}^*\|^2 + n^2 \cdot \frac{1}{n} + 4n \\
&\leq O(\log n) \cdot OPT_{SDP_{TZ_2O}} + 5n \\
&\leq O(\log n) \cdot OPT,
\end{aligned}$$

because $OPT \geq \Omega(n)$, which proves Theorem 3.1.10.

3.5.2 PR-Optimization Problem With Outliers

For this problem, the cost function becomes:

$$\begin{aligned}
&cost(A, F, V, d) \\
&= (n - f) \cdot |A| + |R| \\
&= (n - f) \cdot |A| + \left| \{\{u, v\} \subseteq V \backslash F \mid d(u, v) < \min_{w \in A} d(u, w) + \min_{w \in A} d(v, w) - 1\} \right|.
\end{aligned}$$

$$
\begin{array}{lll}
\min & \displaystyle\sum_{v\in V}(n-f)\cdot\|\vec{x}_v\|^2+\sum_{\{u,v\}\subseteq V}\|\vec{y}_{uv}\|^2 & \\
\text{s.t.} & \|\vec{y}_{uv}\|^2\geq 1-\vec{z}_u\cdot\vec{z}_v & \\
 & \qquad -\displaystyle\sum_{w\in B_u(r)\cup B_v(d(u,v)-r)}\|\vec{x}_w\|^2 & \forall u,v\in V, r\in[0,d(u,v)] \\
 & \displaystyle\sum_{v\in V}\|\vec{z_v}\|^2\leq f & \\
 & \|\vec{x}_v\|^2\leq 1 & \forall v\in V \\
 & \|\vec{y}_{uv}\|^2\leq 1 & \forall u,v\in V \\
 & \|\vec{z}_v\|^2\leq 1 & \forall v\in V
\end{array}
\tag{3.19}
$$

Figure 3.5: SDP relaxation for PR-Optimization Problem With Outliers (SDP_{PR})

We will again use an SDP relaxation. Let $\vec{x}_v$ be a variable which is supposed to be an indicator for whether $v \in A$, let $\vec{y}_{uv}$ be a variable which is supposed to be an indicator for whether $\{u,v\} \in R$, and let $\vec{z}_v$ be a variable which is supposed to be an indicator for whether $v \in F$. We have the relaxation given in Figure 3.5, which is similar to both LP_{PR} and SDP_{TZ_2O}:

Note that this SDP_{PR} is solvable in polynimial time for the same reason that LP_{PR} is solvable: for each pair of (u,v), we can find n different values of r that give all of the distinct constraints.

The rounding algorithm is basically the same as the TZ_2-optimization problem with outliers. We first solve the SDP_{PR} and get an optimal solution $(\vec{x}^*_v, \vec{y}^*_{uv}, \vec{z}^*_v)$. We then use independent randomized rounding to get A, adding each $v \in V$ to A independently with probability $\min\{\frac{6\ln n}{\varepsilon}\cdot\|\vec{x}^*_v\|^2, 1\}$ where ε is a small constant. Then we use threshold rounding to get F, adding each $v \in V$ to F if $\|\vec{z}^*_v\|^2 \geq \frac{1}{1+\varepsilon}$.

3.5.2.1 Bicriteria Approximation: Proof of Theorem 3.1.11

We will prove that the previous algorithm is an $(O(\log n), 1+\varepsilon)$-approximation. It is easy to see that $|F| \leq (1+\varepsilon)f$ because $\sum_{v \in V} \|\vec{z}_v^*\|^2 \leq f$. The proof that the expected cost is at most $O(\log n) \cdot OPT$ is as follows:

Lemma 3.5.2. *If $\|\vec{y}_{uv}^*\|^2 \leq \frac{\varepsilon}{2}$, then the probability that $\{u, v\} \in R$ is at most $\frac{1}{n}$.*

Proof. If $\|\vec{z}_u^*\|^2 \geq \frac{1}{1+\varepsilon}$ or $\|\vec{z}_v^*\|^2 \geq \frac{1}{1+\varepsilon}$, then u or v is in F, so $\{u, v\} \notin R$. Thus we only consider the case that $\|\vec{z}_u^*\|^2 < \frac{1}{1+\varepsilon}$ and $\|\vec{z}_v^*\|^2 < \frac{1}{1+\varepsilon}$, which means $\vec{z}_u^* \cdot \vec{z}_v^* < \frac{1}{1+\varepsilon}$. Since $\|\vec{y}_{uv}^*\|^2 \leq \frac{\varepsilon}{2}$, we have

$$\sum_{w \in B_u(r) \cup B_v(d(u,v)-r)} \|\vec{x}_w^*\|^2 \geq 1 - \frac{\varepsilon}{2} - \frac{1}{1+\varepsilon} \geq \frac{\varepsilon}{3}.$$

Therefore, the probability that $A \cap (B_u(r) \cup B_v(d(u,v) - r)) = \varnothing$ for a specifiic $r \in [0, d(u,v)]$ is at most

$$\prod_{w \in B_u(r) \cup B_v(d(u,v)-r)} (1 - \min\{\frac{6 \ln n}{\varepsilon} \cdot \|\vec{x}_w^*\|^2, 1\}) \leq e^{-\sum_{w \in B_u(v)} \frac{6 \ln n}{\varepsilon} \cdot \|\vec{x}_w^*\|^2} \leq \frac{1}{n^2}$$

□

By using union bound over all the different r we used in our SDP, the probability that there exists an $r \in [0, d(u,v)]$ where $A \cap (B_u(r) \cup B_v(d(u,v) - r)) = \varnothing$ is at most $\frac{1}{n^2} \cdot n = \frac{1}{n}$, which means $d(u,v) < \min_{w \in A} d(u,w) + \min_{w \in A} d(v,w) - 1$ with probability at most $\frac{1}{n}$, so the probability that $\{u, v\} \in R$ is at most $\frac{1}{n}$.

Therefore, let $OPT_{SDP_{PR}}$ denotes the optimal cost of SDP_{PR}, then the expected cost of the rounding algorithm is at most

$$\sum_{v \in V} (n - f) \cdot \frac{3 \ln n}{\varepsilon} \cdot \|\vec{x}_v^*\|^2 + \frac{2}{\varepsilon} \cdot \sum_{u,v \in V} \|\vec{y}_{uv}^*\|^2 + n^2 \cdot \frac{1}{n}$$

$$\leq O(\log n) \cdot OPT_{SDP_{PR}} + n$$

$$\leq O(\log n) \cdot OPT,$$

because $OPT \geq \Omega(n)$, which proves Theorem 3.1.11.

References

Thorup, Mikkel and Uri Zwick (2005). "Approximate distance oracles". In: *Journal of the ACM (JACM)* 52.1, pp. 1–24.

Patrascu, Mihai and Liam Roditty (2010). "Distance oracles beyond the Thorup-Zwick bound". In: *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, pp. 815–823.

Chan, T-H Hubert, Kedar Dhamdhere, Anupam Gupta, Jon Kleinberg, and Aleksandrs Slivkins (2009). "Metric embeddings with relaxed guarantees". In: *SIAM Journal on Computing* 38.6, pp. 2303–2329.

Chan, T-H Hubert, Michael Dinitz, and Anupam Gupta (2006). "Spanners with slack". In: *European Symposium on Algorithms*. Springer, pp. 196–207.

Vazirani, Vijay V (2013). *Approximation algorithms*. Springer Science & Business Media.

Feige, Uriel (1998). "A threshold of ln n for approximating set cover". In: *Journal of the ACM (JACM)* 45.4, pp. 634–652.

Hochbaum, Dorit S (1982). "Heuristics for the fixed cost median problem". In: *Mathematical programming* 22.1, pp. 148–162.

Raz, Ran (1998). "A parallel repetition theorem". In: *SIAM Journal on Computing* 27.3, pp. 763–803.

Ben-Or, Michael, Shafi Goldwasser, Joe Kilian, and Avi Wigderson (1988). "Multi-prover interactive proofs: How to remove intractability assumptions". In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. ACM, pp. 113–131.

Alon, Noga, Oded Goldreich, Johan Håstad, and René Peralta (1992). "Simple Constructions of Almost k-wise Independent Random Variables". In: *Random Structures & Algorithms* 3.3, pp. 289–304.

Raz, Ran and Shmuel Safra (1997). "A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP". In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*. ACM, pp. 475–484.

Chapter 4

Shallow-Light Steiner Networks

4.1 Definitions and Preliminaries

Definition 4.1.1 (SHALLOW-LIGHT STEINER NETWORK)**.** Given a graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}^+$, a length function $l : E \to \mathbb{R}^+$, a distance bound L, and p pairs of vertices $\{s_1, t_1\}, \ldots, \{s_p, t_p\}$. The objective of SLSN is to find a minimum cost subgraph $G' = (V, S)$, such that for every $i \in [p]$, there is a path between s_i and t_i in G' with length less or equal to L.

Let H be the graph with vertex set $\{s_1, \ldots, s_p, t_1, \ldots, t_p\}$ and edge set $\{\{s_1, t_1\}, \ldots, \{s_p, t_p\}\}$. We call H the *demand graph* of the problem. We use $|H|$ to represent the number of edges in H.

In order to distinguish the easy from the hard cases of the SLSN problem with respect to the demand graph, we should first define the problem with respect to a class (set) of demand graphs.

Definition 4.1.2. Given a class $\mathcal{C}$ of graphs. The problem of SHALLOW-LIGHT STEINER NETWORK with restricted demand graph class $\mathcal{C}$ (SLSN$_{\mathcal{C}}$) is the SLSN problem with the additional restriction that the demand graph H of the

problem must be isomorphic to some graph in $\mathcal{C}$.

4.1.1 Results and Techniques

Let $\mathcal{C}_\lambda$ be the class of all demand graphs with at most λ edges, and $\mathcal{C}^*$ be the class of all star demand graphs (there is a central vertex called the root, and every other vertex in the demand graph is adjacent to the root and only the root). Our main result is that these are *precisely* the easy classes: We first prove that SLSN is in XP parameterized by the number of demands (i.e. solvable in $n^{f(p)}$ time for some function f), which immediately implies that $\text{SLSN}_{\mathcal{C}_\lambda}$ can be solved in polynomial time if λ is a constant. Note that $\text{SLSN}_{\mathcal{C}^*}$ is precisely the SLST problem, for which a folklore FPT algorithm exists, thus $\text{SLSN}_{\mathcal{C}^*}$ (while NP-hard) is in FPT for parameter p. We also show that, for any other class $\mathcal{C}$ (i.e., any class which is not just a subset of $\mathcal{C}^* \cup \mathcal{C}_\lambda$ for some constant λ), the problem $\text{SLSN}_{\mathcal{C}}$ is W[1]-hard for parameter p. In other words, if the class of demand graphs includes arbitrarily large non-stars, then the problem is W[1]-hard parameterized by the number of demands.

More formally, we prove the following theorems.

Theorem 4.1.3. *The unit-length arbitrary-cost* SLSN *problem with parameter p is in* XP*, and it can be solved in $n^{O(p^4)}$ time.*

By "unit-length arbitrary-cost" we mean that the length $l(e) = 1$ for all edges $e \in E$, while the cost c is arbitrary. To prove this theorem, we first prove a structural lemma which shows that the optimal solution must be the union of several lowest cost paths with restricted length (these paths may be between steiner nodes, but we show that there cannot be too many). Then we

just need to guess all the endpoints of these paths, as well as all the lengths of these paths. We prove that there are only $n^{O(p^4)}$ possibilities, and the running time is also $n^{O(p^4)}$. The algorithm and proof is in Section 4.2.1.

Theorem 4.1.4. *The unit-length arbitrary-cost* $\text{SLSN}_{\mathcal{C}^*}$ *problem is* FPT *for parameter* p*.*

As mentioned, $\text{SLSN}_{\mathcal{C}^*}$ is exactly the same as SLST, so we use a folklore reduction between SLST and DST to prove this theorem. The detailed proof is in Section 4.2.2 for completeness.

Theorem 4.1.5. *If* $\mathcal{C}$ *is a recursively enumerable class, and* $\mathcal{C} \not\subseteq \mathcal{C}_\lambda \cup \mathcal{C}^*$ *for any constant* λ*, then* $\text{SLSN}_{\mathcal{C}}$ *is* W[1]*-hard for parameter* p*, even in the unit-length and unit-cost case.*

Many W[1]-hardness results for network design problems reduce from the MULTI-COLORED CLIQUE (MCC) problem, and ours are no exception. We reduce from MCC to $\text{SLSN}_{\mathcal{C}'}$, where $\mathcal{C}'$ is a specific subset of $\mathcal{C}$ which has some particularly useful properties, and which we show must exist for any such $\mathcal{C}$. Since $\mathcal{C}' \subseteq \mathcal{C}$, this will imply the theorem. The reduction is in Section 4.3.2.

All of the above results are in the unit-length setting. We extend both our upper bounds and hardness results to handle arbitrary lengths, but with some extra complications. If $p = 1$ (there is only one demand), then with arbitrary lengths and arbitrary costs the SLSN problem is equivalent to the RESTRICTED SHORTEST PATH problem, which is known to be NP-hard (Hassin, 1992). Therefore we can no longer hope for a polynomial time exact solution

when $p = 1$, and thus cannot hope for an FPT algorithm (with parameter p). So we change our notion of "easy" from "solvable in FPT" to "arbitrarily approximable in FPT": we show $(1+\varepsilon)$-approximation algorithms for the easy cases, and prove that there is no $\left(\frac{5}{4}-\varepsilon\right)$-approximation algorithm for the hard cases in $f(p) \cdot poly(n)$ time for any function f.

Theorem 4.1.6. *For any constant $\lambda > 0$, there is a fully polynomial time approximation scheme (*FPTAS*) for the arbitrary-length arbitrary-cost* $\text{SLSN}_{\mathcal{C}_\lambda}$ *problem.*

Theorem 4.1.7. *There is a $(1+\epsilon)$-approximation algorithm in $O(4^p \cdot poly(\frac{n}{\varepsilon}))$ time for the arbitrary-length arbitrary-cost* $\text{SLSN}_{\mathcal{C}^*}$ *problem.*

For both upper bounds, we use basically the same algorithm as in the unit-length arbitrary-cost case, with some changes inspired by the $(1+\varepsilon)$-approximation algorithm for the RESTRICTED SHORTEST PATH problem (Lorenz and Raz, 2001). These results can be found in Section 4.4.

Our next theorem is analogous to Theorem 4.1.5, but since costs are allowed to be arbitrary we can prove stronger hardness of approximation (under stronger assumptions).

Theorem 4.1.8. *Assume that (randomized) Gap-Exponential Time Hypothesis (Gap-ETH, see (Chalermsook et al., 2017)) holds. Let $\varepsilon > 0$ be a small constant, and $\mathcal{C}$ be a recursively enumerable class where $\mathcal{C} \nsubseteq \mathcal{C}_\lambda \cup \mathcal{C}^*$ for any constant λ. Then, there is no $\left(\frac{5}{4}-\varepsilon\right)$-approximation algorithm in $f(p) \cdot n^{O(1)}$ time for* $\text{SLSN}_{\mathcal{C}}$ *for any function f, even in the unit-length and polynomial-cost case.*

Note that this theorem uses a much stronger assumption (Gap-ETH rather than $\text{W}[1] \neq \text{FPT}$), which assumes that there is no (possibly randomized)

algorithm running in $2^{o(n)}$ time that can distinguish whether a 3SAT formula is satisfiable or at most a $(1-\varepsilon)$-fraction of its clauses can be satisfied. This enables us to utilize the hardness result for a generalized version of the MCC problem from (Chitnis, Feldmann, and Manurangsi, 2017), which will allow us to modify our reduction from Theorem 4.1.5 to get hardness of approximation. This result appears in Section 4.5.

4.1.2 Relationship to (Feldmann and Marx, 2016)

As mentioned, our results and techniques are strongly motivated and influenced by the work of Feldmann and Marx (Feldmann and Marx, 2016), who proved similar results in the directed setting. Informally, they showed that DIRECTED STEINER NETWORK is in FPT if the demand graph is transitively equivalent to an "almost-caterpillar", and otherwise it is W[1]-hard. Since "transitively equivalent to an almost-caterpillar" is a complex and subtle class, this showed that the tractability of DSN exhibits interesting behavior. Our results, on the other hand, show that SLSN is extraordinarily hard: there simply are not any algorithms possible for demand graphs that are even a little bit complex, despite the folklore relationships between directed settings and length-bounded settings. Thus our hardness proof is significantly more complicated than the reduction in (Feldmann and Marx, 2016), despite sharing some ideas.

The main case of the hardness reduction of (Feldmann and Marx, 2016) (which, like our reduction, is from MCC) is when the demand graph is a 2-by-k complete bipartite graph (i.e., two stars with the same leaf set). For this case,

their reduction from MCC uses one star to control the choice of edges in the clique and another star to control the choice of vertices in the clique. They set this up so that if there is a clique of the right size then the "edge demands" and the "vertex demands" can be satisfied with low cost by making choices corresponding to the clique, while if no such clique exists then any way of satisfying the two types of demands simultaneously must have larger cost.

The 2-by-k complete bipartite graph is also a hard demand graph in our setting, and the same reduction from (Feldmann and Marx, 2016) can be straightforwardly modified to prove this (this appears as one of our cases). However, we prove that far simpler demand graphs are also hard. Most notably, the "main" case of our proof is when the demand graph is a single star together with one extra edge. Since we have only a single star in our demand graph, we cannot have two "types" of demands (vertex demands and edge demands) in our reduction. So we instead use the star to correspond to "edge demands" and use the single extra edge to simultaneously simulate all of the "vertex demands". This makes our reduction significantly more complicated.

With respect to upper bounds, the algorithm of (Feldmann and Marx, 2016) is quite complex in part due to the complexity of the demand graphs that it must solve. Our hardness results for SLSN imply that we need only concern ourselves with demand graphs that are star or have constant size. The star setting is relatively simple due to a reduction to DST, but it is not obvious how to use any adaptation of (Feldmann and Marx, 2016) (or the earlier (Feldman and Ruhl, 2006)) to handle a constant number of demands for SLSN. Our

algorithm ends up being relatively simple, but requires a structural lemma which was not necessary in the DSN setting.

4.2 Algorithms for the Unit-Length Arbitrary-Cost SLSN

In this section we discuss the "easy" cases of SLSN. We first present an XP algorithm for SLSN in Section 4.2.1. In Section 4.2.2, we describe a reduction from SLSN with star demand graphs to DST, which gives an FPT algorithm.

4.2.1 The XP algorithm

The XP algorithm for Theorem 4.1.3 relies on the following structural lemma , which allows us to limit the structure of the optimal solution. This lemma works not only for the unit-length case, but also for the arbitrary-length case.

Lemma 4.2.1. *In any feasible solution $S \subseteq E$ of the* SLSN *problem, there exists a way to assign a path P_i between s_i and t_i in S for each demand $\{s_i, t_i\} \in H$ such that:*

- *For each $i \in [p]$, the total length of P_i is at most L and there is no cycle in P_i.*
- *For each $i, j \in [p]$ and $u, v \in P_i \cap P_j$, the paths between u and v in P_i and P_j are the same.*

Proof. We give a constructive proof. Let $m = |S|$ and $S = \{e_1, \ldots, e_m\}$. We first want to modify the lengths to ensure that there is always a unique shortest path. Let Δ denote the minimum length difference between any two subsets

of S with different total length, i.e.,

$$\Delta = \min_{A,B \subseteq S, \sum_{e \in A} l(e) \neq \sum_{e \in B} l(e)} \left| \sum_{e \in A} l(e) - \sum_{e \in B} l(e) \right|.$$

We create a new length function g where $g(e_i) = l(e_i) + \Delta \cdot 2^{-i}$. Note that Δ is always non-zero for any S which has at least 2 edges, and the problem is trivial when $|S| = 1$.

We now show that any two paths have different lengths under g. Consider any two different paths P_x and P_y. If $\sum_{e \in P_x} l(e) \neq \sum_{e \in P_y} l(e)$, then without loss of generality we assume $\sum_{e \in P_x} l(e) < \sum_{e \in P_y} l(e)$. Then

$$\sum_{e \in P_x} g(e) \leq \sum_{e \in P_x} l(e) + \sum_{i=1}^{m} \Delta \cdot 2^{-i} < \sum_{e \in P_x} l(e) + \Delta \leq \sum_{e \in P_y} l(e) < \sum_{e \in P_y} g(e). \quad (4.1)$$

Otherwise, if $\sum_{e \in P_x} l(e) = \sum_{e \in P_y} l(e)$, then

$$\sum_{e \in P_x} g(e) - \sum_{e \in P_y} g(e) = \sum_{i : e_i \in P_x} \Delta \cdot 2^{-i} - \sum_{i : e_i \in P_y} \Delta \cdot 2^{-i} \neq 0.$$

Therefore in both cases P_x and P_y have different lengths under g.

For each demand $\{s_i, t_i\} \in H$, we let P_i be the shortest path between s_i and t_i in S under the new length function g. Because any two paths under g have different length, the shortest path between each $\{s_i, t_i\} \in H$ is unique. In addition, because these are shortest paths and edge lengths are positive, they do not contain any cycles.

For each $i \in [p]$, we can see that P_i is also one of the shortest paths between s_i and t_i under original length function l. This is because in equation (4.1) we proved that a shorter path under length function l is still a shorter path under

length function g. Since S is a feasible solution, the shortest path between s_i and t_i in S must have length at most L. Thus for each $i \in [p]$, we have $\sum_{e \in P_i} l(e) \leq L$.

For any two different paths P_i and P_j, let $u, v \in P_i \cap P_j$. If the subpath of P_i between u and v is different from the subpath of P_j between u and v, then by the uniqueness of shortest paths under g we know that either P_i or P_j is not a shortest path (since one of them could be improved by changing the subpath between u and v). This contradicts our definition of P_i and P_j, and hence they must use the same subpath between u and v. □

Lemma 4.2.1 implies that for each two paths P_i and P_j, either they do not share any edge, or they share exactly one (maximal) subpath. Since there are only p demands, the total number of shared subpaths is at most $\binom{p}{2}$. Therefore we can solve the unit-length arbitrary-cost $\text{SLSN}_{\mathcal{C}_\lambda}$ by guessing these subpaths.

The first step of our algorithm is to guess the endpoints Q of these subpaths, and let $Q' = Q \cup \left(\bigcup_{i=1}^{p} \{s_i, t_i\}\right)$. The second step is to guess a set $E' \subseteq \{\{u, v\} \mid u, v \in Q', u \neq v\}$. Intuitively, a pair $\{u, v\} \in E'$ means there is a path between u and v in the optimal solution such that only the endpoints of this path is in Q'. Then we also guess the length $l'(\{u, v\})$ of such path for each $\{u, v\} \in E'$. Finally, we connect each pair of $u, v \in V$ where $\{u, v\} \in E'$ by lowest cost paths with restricted length $l'(\{u, v\})$, check feasibility, and output the optimal solution. The detailed algorithm is in Algorithm 4 in Section 4.2.1.

Claim 4.2.2. *The running time of Algorithm 4 is* $n^{O(p^4)}$.

Algorithm 4 Unit-length arbitrary-cost SLSN

Let $M \leftarrow \sum_{e \in E} c(e)$ and $S \leftarrow E$
for $Q \subseteq V$ where $|Q| \leq p(p-1)$ **do**
  $Q' \leftarrow Q \cup \left(\bigcup_{i=1}^{p} \{s_i, t_i\}\right)$
  for $E' \subseteq \{\{u, v\} \mid u, v \in Q', u \neq v\}$ and $l' : E' \to [L]$ **do**
    $T \leftarrow \varnothing$
    for $\{u, v\} \in E'$ **do**
      $T \leftarrow T \cup \{$the lowest cost path between u and v with length at most $l'(\{u, v\})\}$
      // if such path does not exist, T remains the same
    end for
    if T is a feasible solution and $\sum_{e \in T} l'(e) < M$ **then**
      $M \leftarrow \sum_{e \in T} c(e)$ and $S \leftarrow T$
    end if
  end for
end for
return S

Proof. Clearly there are at most $n^{p(p-1)}$ possibilities for Q, and for each Q there are at most $2^{(p(p-1)+2p)^2}$ possible sets E' and at most $L^{(p(p-1)+2p)^2}$ possible l'. Since we assume unit edge lengths, we can use the Bellman-Ford algorithm to find the lowest cost path within a given length bound in polynomial time. Checking feasibility also takes polynomial time using standard shortest path algorithms. Thus, the running time is at most $n^{p(p-1)} \cdot 2^{(p(p+1))^2} \cdot n^{(p(p+1))^2} \cdot poly(n)$. □

4.2.1.1 Proof of Theorem 4.1.3:

By Claim 4.2.2, the running time of Algorithm 4 is $n^{O(p^4)}$. Now we will prove correctness. The algorithm always returns a feasible solution, because we replace S by T only if T is feasible, and thus S is always a feasible solution. Therefore, we only need to show that this algorithm returns a solution with

cost at most the cost of the optimal solution.

Let the optimal solution be S^*. We assign P_i^* for all $i \in [p]$ as in Lemma 4.2.1. Recall that path P_i^* and P_j^* can share at most one (maximal) subpath for each $i, j \in [p]$ where $i \neq j$. Let Q^* be the endpoint set of the (maximal) subpaths which are shared by some P_i^* and P_j^*, and let $Q'^* = Q^* \cup \bigcup_{i=1}^{p} \{s_i, t_i\}$.

We can see that the optimal solution S^* can be partitioned to a collection of paths by Q^*. We use E'^* to represent whether two vertices in Q'^* are "adjacent" on some path P_i^*: for any $u, v \in Q'^*$ where $u \neq v$, the set E'^* contains $\{u, v\}$ if and only if there exists $i \in [p]$ such that $u, v \in P_i^*$, and there is no vertex $w \in Q'^* \setminus \{u, v\}$ which is in the subpath between u and v in P_i^*. For each $\{u, v\} \in E'^*$, let $P^*_{\{u,v\}}$ be the subpath between u and v on path P_i^*. This is well defined because by Lemma 4.2.1 the subpath is unique. We define $l'^*(\{u, v\})$ as the length of $P^*_{\{u,v\}}$ for each $\{u, v\} \in E'^*$

Note that for any $\{u, v\} \neq \{u', v'\} \in E'^*$, we also know that $P^*_{\{u,v\}}$ and $P^*_{\{u',v'\}}$ are edge-disjoint. To see this, assume that they do share an edge, and let u'' and v'' be the endpoints of the (maximal) shared subpath between $P^*_{\{u,v\}}$ and $P^*_{\{u',v'\}}$. Then u'' and v'' are both in Q'^*, and at least one of them is in $Q'^* \setminus \{u, v\}$ or in $Q'^* \setminus \{u', v'\}$, which contradicts our definition of E'^*.

Since the algorithm iterates over all possibilities for Q, E' and l', there is some iteration in which $Q = Q'^*$, $E' = E'^*$, and $l' \equiv l'^*$. We will show that the algorithm also must find an optimal feasible solution in this iteration.

For each $i \in [p]$, the path P_i^* is partitioned to edge-disjoint subpaths by Q'^*. Let q_i be the number of subpaths, and let the endpoints be $s_i =$

$v_{i,0}, v_{i,1}, \ldots, v_{i,q_i-1}, v_{i,q_i} = t_i$. We further let these subpaths be

$$P^*_{\{s_i,v_{i,1}\}}, P^*_{\{v_{i,1},v_{i,2}\}}, \ldots, P^*_{\{v_{i,q_i-1},t_i\}}.$$

By the definition of l'^*, for each $j \in [q_i]$, there must be a path between $v_{i,j-1}$ and $v_{i,j}$ with length at most $l'^*(\{v_{i,j-1}, v_{i,j}\})$ in graph G. Thus after the algorithm visited $\{v_{i,j-1}, v_{i,j}\} \in E'^*$, the edge set T must contains a path between u and v with length at most $l'^*(\{v_{i,j-1}, v_{i,j}\})$. Therefore we know that the edge set T in this iteration contains a path between s_i and t_i with length $\sum_{j=1}^{q_i} l'^*(\{v_{i,j-1}, v_{i,j}\}) \leq L$, and thus it is a feasible solution.

Let $MinCost(u, v, d)$ be the lowest cost for a path between u and v with distance at most d in graph G, then the total cost of this solution is

$$\sum_{\{u,v\} \in E'^*} MinCost(u, v, l'^*(\{u, v\})).$$

Moreover, for each $\{u, v\} \in E'^*$ and $\{u', v'\} \in E'^*$ with $\{u, v\} \neq \{u', v'\}$, the paths $P^*_{\{u,v\}}$ and $P^*_{\{u',v'\}}$ are edge-disjoint, and each $P^*_{\{u,v\}}$ has cost at least $MinCost(u, v, l'^*(\{u, v\}))$. Thus the cost of the optimal solution is at least $\sum_{\{u,v\} \in E'^*} MinCost(u, v, l'^*(\{u, v\}))$, and so the algorithm outputs an optimal solution and it runs in polynomial time. □

Corollary 4.2.3. *The arbitrary-length unit-cost* SLSN *problem with parameter p is in* XP.

Proof. We can use the same technique, but instead of guessing the length l' we guess the cost c', and then find shortest path under cost bound c'. We can also use Bellman-Ford algorithm in this step. □

4.2.2 Star Demand Graphs ($\mathbf{SLSN}_{\mathcal{C}^*}$)

We do a reduction from $\text{SLSN}_{\mathcal{C}^*}$ to the DST problem. This is essentially folklore. We include it here for completeness.

Definition 4.2.4 (DIRECTED STEINER TREE)**.** Given a directed graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}^+$, a root s, and p vertices $t_1, \ldots, t_p$, the objective of the DST problem is to find a minimum cost subgraph $G' = (V, S)$, such that for every $i \in [p]$, there is a path from s to t_i in G'.

Theorem 4.2.5 (Feldman and Ruhl, 2006)**.** *There is an* FPT *algorithm for the* DST *problem for parameter p.*

4.2.2.1 Proof of Theorem 4.1.4:

Let $(G = (V, E), c, l \equiv 1, \{\{s_1, t_1\}, \ldots, \{s_p, t_p\}\}, L)$ be a unit-length arbitrary-cost SLSN instance with restricted demand graph class $\mathcal{C}^*$. Since $\mathcal{C}^*$ is the class of stars, we let $s = s_1 = s_2 = \ldots = s_p$.

For the reduction, we first create a $(L+1)$-layered graph G', where each layer has $|V|$ vertices. Let $v^{(i)}$ represent the vertex $v \in V$ in layer i. Then for each $i \in [L]$ and $u, v \in V$, we add an edge from $u^{(i-1)}$ to $v^{(i)}$ if $\{u, v\} \in E$, and we give this edge cost $c'(u^{(i-1)}, v^{(i)}) = c(u, v)$. For each $i \in [L]$ and each $v \in V$, we also add an edge $(v^{(i-1)}, v^{(i)})$ with cost $c'(v^{(i-1)}, v^{(i)}) = 0$.

This gives us an instance $(G', c', s^{(0)}, t_1^{(L)}, \ldots, t_p^{(L)})$ of DST. Since this reduction clearly takes only polynomial time (since $L \leq n$ due to the unit-length setting), the only thing left is to show that the two instances have the same optimal cost.

Let S be the optimal solution of our starting SLSN instance. Let $d_s(v)$ be the distance between s and v in S. We can construct a solution S' for the DST instance of cost at most $c(S)$. First, for each $i \in [L]$ and $\{u, v\} \in S$ with $d_s(u) + 1 = d_s(v)$, we add $(u^{(d_s(u))}, v^{(d_s(v))})$ to S'. Then, for each $j \in [p]$ and $i = d_s(t_j), \ldots, L$, we add $(t_j^{(i-1)}, t_j^{(i)})$ to S'. Note that the cost of S' is at most the cost of S, since every non-zero cost edge in S' corresponds to a different edge in S with the same cost. S' is also a feasible solution, because for every $i \in [p]$ there is a path $s - v_{i,1} - \ldots - v_{i,d_s(t_i)-1} - t_i$ in S with length at most L, such that $d_s(v_{i,j}) = j$ for each $j \in [d_s(t_i) - 1]$, and thus S' contains path $s^{(0)} - v_{i,1}^{(1)} - \ldots - v_{i,d_s(t_i)-1}^{(d_s(t_i)-1)} - t_i^{(d_s(t_i))} - \ldots - t_i^{(L)}$.

Now let S' be the optimal solution of the DST instance. We can construct a solution S for our original SLSN$_{\mathcal{C}^*}$ instance as follows: for any $u, v \in V$ where $u \neq v$, we add $\{u, v\}$ to S if there exists an i such that $(u^{(i-1)}, v^{(i)}) \in S'$. Clearly the cost of S is at most the cost of S' because every edge in S corresponds to a different edge in S' with the same cost. S is also a feasible solution, since for every $i \in [p]$ there is a path $s^{(0)} = v_{i,0}^{(0)} - v_{i,1}^{(1)} - \ldots - v_{i,L-1}^{(L-1)} - v_{i,L}^{(L)} = t_i^{(L)}$ in S', and thus S contains path $s - v_{i,1} - \ldots - v_{i,L-1} - t_i$ with length at most L. Notice that there may be $j \in [L]$ such that $v_{i,j} = v_{i,j-1}$, but this only decreases the length and has no effect on cost.

Therefore, the two instances have the same optimal cost. Combining this with Theorem 4.2.5 allows us to get an FPT algorithm for the unit-length arbitrary-cost SLSN$_{\mathcal{C}^*}$ by first reducing to DST and then using the algorithm from Theorem 4.2.5. □

4.3 W[1]-Hardness for the Unit-Length Unit-Cost SLSN

In this section we prove our main hardness result, Theorem 4.1.5. We begin with some preliminaries, then give our reduction and proof.

4.3.1 Preliminaries

We prove Theorem 4.1.5 by constructing an FPT reduction from the MULTI-COLORED CLIQUE (MCC) problem to the unit-length unit-cost SLSN$_{\mathcal{C}}$ problem for any $\mathcal{C} \not\subseteq \mathcal{C}_\lambda \cup \mathcal{C}^*$. We begin with the MCC problem.

Definition 4.3.1 (MULTI-COLORED CLIQUE)**.** Given a graph $G = (V, E)$, a number $k \in \mathbb{N}$ and a coloring function $c : V \to [k]$. The objective of the MCC problem is to determine whether there is a clique $T \subseteq V$ in G with $|T| = k$ where $c(x) \neq c(y)$ for all $x, y \in T$.

For each $i \in [k]$, we define $C_i = \{v \in V : c(v) = i\}$ to be the vertices of color i. We can assume that the graph does not contain edges where both endpoints have the same color, since those edges do not affect the existence of a multi-colored clique. It has been proven that the MCC problem is W[1]-complete.

Theorem 4.3.2 (Fellows et al., 2009)**.** *The* MCC *problem is* W[1]*-complete with parameter k.*

We first define a few important classes of graphs. These are the major classes that fall outside of $\mathcal{C}^* \cup \mathcal{C}_\lambda$, so we will need to be able to reduce MCC

to SLSN where the demand graphs are in these classes, and then this will allow us to prove the hardness for general $C \not\subseteq \mathcal{C}^* \cup \mathcal{C}_\lambda$. For every $k \in \mathbb{N}$, we define the following graph classes. Each of the first four classes is just one graph up to isomorphism, but classes 5 and 6 are sets of graphs, so we use the notation $\mathcal{H}$ instead of H for these classes. Note that each of the first three classes is just a star with an additional edge, so we use * to make this clear.

1. $H^*_{k,0}$: a star with $k(k-1)$ leaves and an edge with both endpoints *not* in the star.

2. $H^*_{k,1}$: a star with $(k(k-1)+1)$ leaves and an edge $\{u, v\}$ where u is a leaf of the star and v is not in the star.

3. $H^*_{k,2}$: a star with $(k(k-1)+2)$ leaves, and an edge $\{u, v\}$ where both u and v are leaves of the star.

4. $H_{k,k}$: $k(k-1)+1$ edges where all endpoints are different (i.e., a matching of size $k(k-1)+1$).

5. $\mathcal{H}_{2,k}$: the class of graphs that have exactly $k(k-1)+2$ vertices, and contain a 2 by $k(k-1)$ complete bipartite subgraph (not necessarily an induced subgraph).

6. $\mathcal{H}_k$: the class of graphs that contain at least one of the graphs in previous five classes as an induced subgraph.

We first prove the following lemma.

Lemma 4.3.3. *For any $k \geq 2$, if a graph H is not a star and H has at least $8k^{10}$ edges, then $H \in \mathcal{H}_k$, and we can find an induced subgraph which is isomorphic to a graph in $\{H^*_{k,0}, H^*_{k,1}, H^*_{k,2}, H_{k,k}\} \cup \mathcal{H}_{2,k} \cup \mathcal{H}_k$ in $poly(|H|)$ time.*

Proof. We give a constructive proof. We first claim that either there is a vertex in H which has degree at least $2k^4$ or there is an induced matching in H of size k^2. Suppose that all vertices have degree less than $2k^4$. Then we can create an induced matching by adding an arbitrary edge $\{u, v\} \in H$ to a edge set M, removing all vertices that are adjacent to either u or v from H, and repeating until there are no more edges in H. In each iteration we reduce the total number of edges by at most $2 \cdot 2k^4 \cdot 2k^4$, thus $|M| \geq \frac{8k^{10}}{8k^8} = k^2$. Since when we add an edge $\{u, v\}$ we also remove all vertices adjacent to u or v, every future edge we add to M will have endpoints which are not adjacent to u or v, and thus M is an induced matching of H with size k^2.

If H has an induced matching of size k^2, then $H \in \mathcal{H}_k$ because it contains $H_{k,k}$ as an induced subgraph, and thus we are done.

Otherwise, H has a vertex s with degree at least $2k^4$. Let S be the neighbors of s. If there is any vertex other than s that is adjacent to at least $k(k-1)$ vertices in S, then H contains a 2 by $k(k-1)$ complete bipartite subgraph, so it contains an induced subgraph $H' \in \mathcal{H}_{2,k}$ and thus is in $\mathcal{H}_k$.

So suppose that there is no vertex other than s that is adjacent to at least $k(k-1)$ vertices in S. Consider the case that there is no edge between any pair of vertices in S; then, because H is not a star, there must be an edge $\{u, v\} \in H$ with at least one of u, v not in $S \cup \{s\}$. Since both u and v are adjacent to at most $k(k-1)$ vertices in S, there are at least $k^4 - 2 \cdot k(k-1) \geq k(k-1)$

vertices in S that are not adjacent to either u or v. Let the set of these vertices be T. Then the induced subgraph on vertex set $T \cup \{s, u, v\}$ is either $H^*_{k,0}$ or $H^*_{k,1}$, depending on whether $\{u, v\} \cap T$ is an empty set.

Now the only remaining case is that there is at least one edge in H with both endpoints in S. In this case, we can find $H^*_{k,2}$ as an induced subgraph as follows: We first let $S_0 = S$. Then, in each iteration t we let v_t be a vertex in S_{t-1} that is adjacent to the fewest number of other vertices in S_{t-1}. We add v_t to the vertex set T, and then delete v_t and all the vertices in S_{t-1} that are adjacent to v_t to get S_t. This process repeats until we have $|T| = k(k-1)$.

We can use induction to show that, after each iteration $t \leq k(k-1)$, there is always at least one edge in H where both endpoints are in S_t. The base case is $t = 0$, where such an edge clearly exists. Assume the claim holds for iteration $t-1$, consider the iteration $t \leq k(k-1)$. If v_t is not adjacent to any other vertex in S_{t-1}, then removing v_t does not affect the fact that there is at least one edge left, and thus the claim still holds. Otherwise, v_t is adjacent to at least one vertex in S_{t-1}. Thus, each vertex in S_{t-1} must be adjacent to at least one vertex in S_{t-1}. Since there is no vertex other than s which is adjacent to at least $k(k-1)$ vertices in S, we know that at most k^2 vertices are deleted in each iteration, and thus there are still at least $2k^4 - k^2 \cdot k(k-1) \geq k^4$ vertices in S_{t-1}. Because removing v_t and its neighbors can only affect the degree of at most $k^2(k-1)^2$ vertices in S_{t-1}, there must still be an edge left between the vertices in S_t.

Let $\{u, v\}$ be one of the edges in H where both endpoints are in S_t, then the induced subgraph on vertex set $T \cup \{s, u, v\}$ is $H^*_{k,2}$. Thus $H \in \mathcal{H}_k$.

It is easy to see that all the previous steps directly find an induced subgraph which is isomorphic to a graph in $\{H^*_{k,0}, H^*_{k,1}, H^*_{k,2}, H_{k,k}\} \cup \mathcal{H}_{2,k} \cup \mathcal{H}_k$ and takes polynomial time, thus the lemma is proved. □

4.3.2 Reduction

In this subsection, we will prove the following reduction theorem.

Theorem 4.3.4. *Let $(G = (V, E), c)$ be an* MCC *instance with parameter k, and let $H \in \mathcal{H}_k$ be a demand graph. Then a unit-length unit-cost* SLSN *instance (G', L) with demand graph H can be constructed in $poly(|V||H|)$ time, and there exists a function g (computable in time $poly(|H|)$) such that the* MCC *instance has a clique with size k if and only if the* SLSN *instance has a solution with cost $g(H)$.*

In order to prove this theorem, we first introduce a construction for any demand graph $H \in \{H^*_{k,0}, H^*_{k,1}, H^*_{k,2}, H_{k,k}\} \cup \mathcal{H}_{2,k}$, and then use the instances constructed in these cases to construct the instance for general $H \in \mathcal{H}_k$.

The construction for $H \in \mathcal{H}_{2,k}$ is similar to (Feldmann and Marx, 2016), which proves the W[1]-hardness of the DSN problem. We change all the directed edges in their construction to undirected, and add some edges and dummy vertices. This construction is presented in Section 4.3.2.3. To handle $H^*_{k,0}$, $H^*_{k,1}$, $H^*_{k,2}$, and $H_{k,k}$, we need to change this basic construction due to the simplicity of the demand graphs. Because the constructions for these four graphs are quite similar, we first introduce the construction for $H^*_{k,0}$ in Section 4.3.2.1, and then show how to modify it for $H^*_{k,1}$, $H^*_{k,2}$, and $H_{k,k}$ in Section 4.3.2.2.

4.3.2.1 Case 1: $H^*_{k,0}$

Given an MCC instance $(G = (V, E), c)$ with parameter k, we create a unit-length and unit-cost SLSN instance (G', L) with demand graph $H^*_{k,0}$ as follows.

We first create a graph G^*_k with integer edge lengths (we will later replace all non-unit length edges by paths). See Figure 4.1 for an overview of this graph. The vertex set V^*_k contains 6 layers of vertices and another group of vertices. The first layer V_1 is just a root r. The second layer V_2 contains a vertex $z_{\{i,j\}}$ for each $1 \leq i < j \leq k$, so there are $\binom{k}{2}$ vertices. The third layer V_3 contains a vertex z_e for each $e \in E$, so there are $|E|$ vertices. The fourth layer V_4 contains a vertex $x_{v,j}$ for each $v \in V$ and $j \in [k]$ with $j \neq c(v)$, so there are $|V| \cdot (k-1)$ vertices. The fifth layer V_5 again contains a vertex $x'_{v,j}$ for each $v \in V$ and $j \in [k]$ with $j \neq c(v)$. The sixth layer V_6 contains a vertex $l_{i,j}$ for each $i, j \in [k]$ where $i \neq j$, so there are $k(k-1)$ vertices. Finally, we have a vertex y_i for $i = 0, \ldots, k$, so there are $k+1$ vertices in the set V_y.

Let $f_i : \mathbb{N} \to \mathbb{N}$ be the function defined by $f_i(j) = j+1$ if $j+1 \neq i$ and $f_i(j) = j+2$ if $j+1 = i$. This function gives the next integer after j, but skips i. Let $f^t_i(j) = f_i(f_i(\ldots f_i(j)))$ denote this function repeated t times. Recall that $C_i = \{v \in V : c(v) = i\}$. The edge set E^*_k contains following edges, with lengths as indicated:

- $E_1 = \{\{r, z_{\{i,j\}}\} \mid 1 \leq i < j \leq k\}$, each edge in E_1 has length 2.
- $E_2 = \{\{z_{\{c(u),c(v)\}}, z_e\} \mid e = \{u, v\} \in E\}$, each edge in E_2 has length 1.
- $E_3 = \{\{z_e, x_{u,c(v)}\} \mid e = \{u, v\} \in E\}$, each edge in E_3 has length $2k^2 - 2$. Note that if $\{z_e, x_{u,c(v)}\} \in E_3$, then $\{z_e, x_{v,c(u)}\} \in E_3$

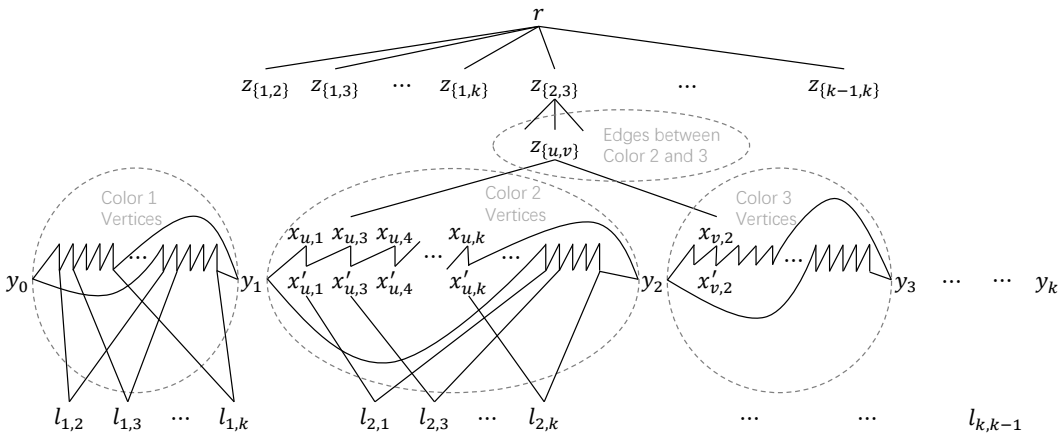

Figure 4.1: G_k^*

- $E_4 = \{\{x_{v,j}, x'_{v,j}\} \mid v \in V, j \neq c(v)\}$, each edge in E_4 has length 1.
- $E_5 = \{\{x'_{v,j}, l_{c(v),j}\} \mid v \in V, j \neq c(v)\}$, each edge in E_5 has length $2k^2 - 2$.
- $E_{yx} = \{\{y_{i-1}, x_{v,f_i(0)}\} \mid i \in [k], v \in C_i\}$, each edge in E_{yx} has length 4.
- $E_{xx} = \{\{x'_{v,j}, x_{v,f_{c(v)}(j)}\} \mid v \in V, j \in [k] \setminus \{c(v), f_{c(v)}^{k-1}(0)\}\}$, each edge in E_{xx} has length 3.
- $E_{xy} = \{\{x'_{v,f_i^{k-1}(0)}, y_i\} \mid i \in [k], v \in C_i\}$, each edge in E_{xy} has length 3.

Let G' be the graph obtained from G_k^* by replacing each edge $e \in E_k^*$ by a $length(e)$-hop path. We create an instance of SLSN on G' by setting the demands to be $\{r, l_{i,j}\}$ for all $i, j \in [k]$ where $i \neq j$, as well as $\{y_0, y_k\}$. Note that these demands form a star with $k(k-1)$ leaves and an edge with both endpoints not in the star, so it is isomorphic to $H_{k,0}^*$. We set the distance bound L to be $4k^2$.

This construction clearly takes $poly(|V||H_{k,0}^*|)$ time. Let $g(H_{k,0}^*) = 4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k$, which is clearly computable in $poly(H_{k,0}^*)$ time. We will first

prove the easy direction in the correctness of the construction.

Lemma 4.3.5. *If there is a multi-colored clique of size k in G, then there is a solution S for the* SLSN *instance* (G', L) *with demand graph* $H^*_{k,0}$*, and the total cost of S is* $g(H^*_{k,0})$.

Proof. Let $v_1, \ldots, v_k$ be a multi-colored clique of size k in G, where $v_i \in C_i$ for all $i \in [k]$. We create a feasible solution S to our SLSN instance, which contains following paths in G' (i.e., edges in G^*_k):

- $\{r, z_{\{i,j\}}\}$ for each $1 \leq i < j \leq k$. The total cost of these edges is $2 \cdot \binom{k}{2} = k^2 - k$.
- $\{z_{\{i,j\}}, z_{\{v_i,v_j\}}\}$ for each $1 \leq i < j \leq k$. The total cost of these edges is $\binom{k}{2} = \frac{k^2-k}{2}$.
- $\{z_{\{v_i,v_j\}}, x_{v_i,j}\}$ and $\{z_{\{v_i,v_j\}}, x_{v_j,i}\}$ for each $1 \leq i < j \leq k$. The total cost of these edges is $2 \cdot (2k^2 - 2) \cdot \binom{k}{2} = 2k^4 - 2k^3 - 2k^2 + 2k$.
- $\{x_{v_i,j}, x'_{v_i,j}\}$ for each $i, j \in [k]$ where $i \neq j$. The total cost of these edges is $2 \cdot \binom{k}{2} = k^2 - k$.
- $\{x'_{v_i,j}, l_{i,j}\}$ for each $i, j \in [k]$ where $i \neq j$. The total cost of these edges is $2 \cdot (2k^2 - 2) \cdot \binom{k}{2} = 2k^4 - 2k^3 - 2k^2 + 2k$.
- $\{y_{i-1}, x_{v_i,f_i(0)}\}$ for each $i \in [k]$. The total cost of these edges is $4k$.
- $\{x'_{v_i,j}, x_{v_i,f_i(j)}\}$ for each $i \in [k]$ and $j \in [k] \setminus \{i, f_i^{k-1}(0)\}$. The total cost of these edges is $3 \cdot k(k-2) = 3k^2 - 6k$.
- $\{x'_{v_i,f_i^{k-1}(0)}, y_i\}$ for each $i \in [k]$. The total cost of these edges is $3k$.

Therefore, the total cost is $k^2 - k + \frac{k^2-k}{2} + 2k^4 - 2k^3 - 2k^2 + 2k + k^2 - k + 2k^4 - 2k^3 - 2k^2 + 2k + 4k + 3k^2 - 6k + 3k = 4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k = g(H^*_{k,0})$.

Now we show the feasibility of this solution. For each $i, j \in [k]$ where $i \neq j$, the path between r and $l_{i,j}$ is $r - z_{\{i,j\}} - z_{\{v_i,v_j\}} - x_{v_i,j} - x'_{v_i,j} - l_{i,j}$. The length of this path is $2 + 1 + 2k^2 - 2 + 1 + 2k^2 - 2 = 4k^2$, thus it is a feasible path.

The path between y_0 and y_k is $y_0 - x_{v_1,2} - x'_{v_1,2} - x_{v_1,3} - x'_{v_1,3} - \ldots - x_{v_1,k} - x'_{v_1,k} - y_1 - x_{v_2,1} - x'_{v_2,1} - x_{v_2,3} - x'_{v_2,3} - \ldots - y_2 - \ldots - y_k$. The length of this path is $(4 + 1 \cdot (k-1) + 3 \cdot (k-2) + 3) \cdot k = 4k^2$, thus it is a feasible path. □

For the other direction, we begin the proof with a few claims. We first show that the only feasible way to connect r and $l_{i,j}$ is to pick one edge between every two adjacent layers. We can also see in Figure 4.1 that for each $i \in [k]$, there are $|C_i|$ disjoint "zig-zag" paths between y_{i-1} and y_i, and each path corresponds to a vertex with color i. We will also show that the only feasible way to connect y_0 and y_k is to pick one zig-zag path between each y_{i-1} and y_i. From these claims we can then prove that, if the cost of the optimal solution is at most $g(H^*_{k,0})$, then there is a multi-colored clique in G.

Claim 4.3.6. *For all $i, j \in [k]$ where $i \neq j$, any path $P_{i,j}$ between r and $l_{i,j}$ with length at most $4k^2$ must be of the form $r - z_{\{i,j\}} - z_{\{u,v\}} - x_{u,j} - x'_{u,j} - l_{i,j}$, where $u \in C_i$, $v \in C_j$ and $\{u, v\} \in E$.*

Proof. We can see that G^*_k is a 6-layer graph with a few additional paths between the fourth layer and the fifth layer. Thus $P_{i,j}$ must contain at least one edge between each two adjacent layers. From the construction of G^*_k, all the edges between two adjacent layers have the same length. If we sum up the

length from r to the fourth layer plus the length from the fifth layer to $l_{i,j}$, it is already $2 + 1 + 2k^2 - 2 + 2k^2 - 2 = 4k^2 - 1$. Thus, between the fourth layer and the fifth layer we can only choose one length 1 edge.

We know that the vertex in the fifth layer must adjacent to $l_{i,j}$, so it must be $x'_{u,j}$ for some $u \in C_i$. Thus, the edge between the fourth layer and the fifth layer must be $\{x_{u,j}, x'_{u,j}\}$, because this is the only length 1 edge adjacent to $x'_{u,j}$. In addition, the only way to go from r to $x_{u,j}$ with one edge per layer is to pass through vertex $z_{\{i,j\}}$ and $z_{\{u,v\}}$ for some $v \in C_j$ and $\{u, v\} \in E$. Therefore $P_{i,j}$ must correspond to an edge $\{u, v\} \in E$ where $u \in C_i$ and $v \in C_j$, and it has form $r - z_{\{i,j\}} - z_{\{u,v\}} - x_{u,j} - x'_{u,j} - l_{i,j}$. □

Claim 4.3.7. *Any path P_y between y_0 and y_k with length at most $4k^2$ does not contain any edge in $E_1 \cup E_2 \cup E_3 \cup E_5$.*

Proof. We prove the claim by contradiction. If P_y contains an edge in $E_1 \cup E_2 \cup E_3 \cup E_5$, it must contain at least two edges with length $2k^2 - 2$ (one edge to go out of the fourth and the fifth layer, and another one to go back). Since any edge which has endpoint y_0 has length 4 and any edge which has endpoint y_k has length 3, the total length $2 \cdot (2k^2 - 2) + 4 + 3 = 4k^2 + 3$ already exceeds the length bound $4k^2$, giving a contradiction. □

Claim 4.3.8. *Any path P_y between y_0 and y_k with length at most $4k^2$ can be divided to k subpaths as follows. For each $i \in [k]$, there is a subpath P_{v_i} between y_{i-1} and y_i with length $4k$, of the form $y_{i-1} - x_{v_i,f_i(0)} - x'_{v_i,f_i(0)} - x_{v_i,f_i^2(0)} - x'_{v_i,f_i^2(0)} - \cdots - x_{v_i,f_i^{k-1}(0)} - x'_{v_i,f_i^{k-1}(0)} - y_i$, where $v_i \in C_i$.*

Proof. Since we have Claim 4.3.7, it suffices to consider the edge set $E_4 \cup E_{yx} \cup$

$E_{xx} \cup E_{xy}$. We can see that $E_4 \cup E_{yx} \cup E_{xx} \cup E_{xy}$ can be partitioned to $k|V|$ paths, where for each $i \in [k]$ and each $v \in C_i$, there is a path P_v which connects y_{i-1} and y_i with length $4k$. The path is $y_{i-1} - x_{v,f_i(0)} - x'_{v,f_i(0)} - x_{v,f_i^2(0)} - x'_{v,f_i^2(0)} - \ldots - x_{v,f_i^{k-1}(0)} - x'_{v,f_i^{k-1}(0)} - y_1$. We can see that these paths are vertex disjoint except for the endpoints $y_0, y_1, \ldots, y_k$.

Therefore, the only way to go from y_0 to y_k is by passing through $y_0, y_1, \ldots, y_k$ one-by-one. Thus, for each $i \in [k]$, P_y must contain a subpath P_{v_i} where $v_i \in C_i$. Because each of these subpaths has length $4k$, the total cost is already $4k \cdot k = 4k^2$, which is exactly the length bound. Therefore, P_y can not contain any other edge, which proves the lemma. □

Now, we can prove the other direction in the correctness of the construction.

Lemma 4.3.9. *Let S be an optimal solution for the* SLSN *instance (G', L) with demand graph $H^*_{k,0}$. If S has cost at most $g(H^*_{k,0}) = 4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k$, then there is a multi-colored clique of size k in G.*

Proof. For each $i, j \in [k]$ with $i \neq j$, let $P_{i,j}$ be a (arbitrarily chosen) path in S which connects r and $l_{i,j}$ with length at most $L = 4k^2$. Let $\mathcal{P} = \{P_{i,j} \mid i, j \in [k], i \neq j\}$ be the set of all these paths. We also let P_y be a (arbitrary) path in S of length at most L which connects y_0 and y_k.

From Claim 4.3.8, P_y can be divided to k subpaths, each of which corresponds to a vertex v_i. We will show that $v_1, \ldots, v_k$ form a clique in G (i.e., for each $1 \leq i < j \leq k$, we have $\{v_i, v_j\} \in E$).

We first prove that these paths must share certain edges due to the cost

bound of the optimal solution. From Claim 4.3.6, we know that each $P_{i,j}$ costs exactly $2+1+2k^2-2+1+2k^2-2 = 4k^2$. In addition, from the form of $P_{i,j}$ we can also see that these paths are almost disjoint, except that $P_{i,j}$ and $P_{j,i}$ may share a length 2 edge $\{r, z_{\{i,j\}}\} \in E_1$ and a length 1 edge $\{z_{\{i,j\}}, z_e\} \in E_2$. Therefore, in order to satisfy the demands between r and all of the $l_{i,j}$'s, the total cost of the edges in $S \cap (E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5)$ is at least $4k^2 \cdot k(k-1) - \binom{k}{2} \cdot (2+1) = 4k^4 - 4k^3 - \frac{3}{2}k^2 + \frac{3}{2}k$, even if every $P_{i,j}$ and $P_{j,i}$ do share edge $\{r, z_{\{i,j\}}\}$ and edge $\{z_{\{i,j\}}, z_e\}$.

We now calculate the cost of the edges in $S \cap (E_{yx} \cup E_{xx} \cup E_{xy})$. From Claim 4.3.8, the total cost of edges in $P_y \cap (E_{yx} \cup E_{xx} \cup E_{xy})$ is at least $(4 + 3 \cdot (k-1) + 3) \cdot k = 3k^2 + k$. Thus, the total cost is already at least $\left(4k^4 - 4k^3 - \frac{3}{2}k^2 + \frac{3}{2}k\right) + (3k^2 + k) = 4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k = g(H^*_{k,0})$, so S cannot contain any edge which has not been counted yet.

Therefore, every edge in $P_y \cap E_4$ must appear in some path in $\mathcal{P}$. In fact, by the form of the paths in $\mathcal{P}$, we can see that for each $i, j \in [k]$ where $i \neq j$, the edge $\{x_{v_i,j}, x'_{v_i,j}\} \in P_y \cap E_4$ can only appear in path $P_{i,j}$, rather than any other $P_{i',j'}$. Thus $x_{v_i,j}$ is in path $P_{i,j}$, and similarly $x_{v_j,i}$ is in path $P_{j,i}$. Recall that $P_{i,j}$ and $P_{j,i}$ must share an edge $\{z_{\{i,j\}}, z_e\}$ for some $e \in E$ because of the cost bound, and $z_{\{v_i,v_j\}}$ is the only vertex which adjacent to both $x_{v_i,j}$ and $x_{v_j,i}$, we can see that e can only be $\{v_i, v_j\}$. Therefore $\{v_i, v_j\} \in E$, which proves the lemma. □

4.3.2.2 Case 2, 3, and 4:

Cases 2, 3, and 4 are basically the same as Case 1, so we discuss them in the same subsection.

Case 2: $H^*_{k,1}$

We use the same G^*_k, G', and L in the construction of the SLSN instance for demand graph $H^*_{k,0}$, and also set $g(H^*_{k,1}) = 4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k$. The only difference is the demand graph. Besides the demand of $\{r, l_{i,j}\}$ for all $i, j \in [k]$ where $i \neq j$, and $\{y_0, y_k\}$, there is a new demand $\{r, y_0\}$. Clearly this new demand graph is a star with $(k(k-1)+1)$ leaves, and an edge in which exactly one of the endpoints is a leaf of the star, so it is isomorphic to $H^*_{k,1}$.

Assume there is a multi-colored clique of size k in G. The paths connecting previous demands in the solution of the SLSN instance are the same as Case 1. The path between r and y_0 is $r - z_{\{1,2\}} - z_{\{v_1,v_2\}} - x_{v_1,2} - y_0$. All the edges in this path are already in the previous paths, so the cost remains the same. The length of this path is $2 + 1 + 2k^2 - 2 + 4 = 2k^2 + 5 < 4k^2$, which satisfies the length bound.

Assume there is a solution for the SLSN instance $(G', L, H^*_{k,1})$ with cost $4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k$. The proof that there exists a multi-colored clique of size k in G is the same as Case 1.

Case 3: $H^*_{k,2}$

As in Case 2, only the demand graph changes. The new demand graph

is the same as in Case 2 but again with a new demand $\{r, y_k\}$. Since $\{r, y_0\}$ was already a demand, our new demand graph is a star with $(k(k-1)+2)$ leaves (the $l_{i,j}$'s and y_0 and y_k), and an edge between two of its leaves (y_0 and y_k), which is isomorphic to $H^*_{k,2}$.

Assume there is a multi-colored clique of size k in G. The paths connecting previous demands in the solution of the SLSN instance are the same as Case 2. The path between r and y_k is $r - z_{\{k-1,k\}} - z_{\{v_{k-1},v_k\}} - x_{v_k,k-1} - y_k$. All the edges in this path are already in the previous paths, so the cost stays the same. The length of this path is $2 + 1 + 2k^2 - 2 + 4 = 2k^2 + 5 < 4k^2$, which satisfies the length bound.

Assume there is a solution for the SLSN instance $(G', L, H^*_{k,2})$ with cost $4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k$. The proof that there exists a multi-colored clique of size k in G is the same as Case 1.

Case 4: $H_{k,k}$

In order to get $H_{k,k}$ as our demand graph, we have to slightly change the construction from Case 1. We still first make a weighted graph $G_{k,k} = (V_{k,k}, E_{k,k})$ and then transform it to the unit-length unit-cost graph G'. For the vertex set $V_{k,k}$, we add another layer of vertices $V_0 = \{l'_{i,j} \mid i, j \in [k], i \neq j\}$ to V^*_k before the first layer V_1. For the edge set $E_{k,k}$, we include all the edges in E^*_k, but change the length of edges in E_1 to length 1. We also add another edge set $E_0 = \{\{l'_{i,j}, r\} \mid i, j \in [k], i \neq j\}$. Each edge in E_0 has length 1.

The demands are $\{l'_{i,j}, l_{i,j}\}$ for each $i, j \in [k]$ where $i \neq j$, as well as $\{y_0, y_k\}$. This is a matching of size $k(k-1)+1$, which is isomorphic to $H_{k,k}$. We still

set the length bound to be $L = 4k^2$, and set $g(H_{k,k}) = 4k^4 - 4k^3 + 2k^2 + 2k$.

If there is a multi-colored clique of size k in G, the construction for the solution in G' is similar to Case 1. For each $i, j \in [k]$ where $i \neq j$, the path between $l'_{i,j}$ and $l_{i,j}$ becomes $l'_{i,j} - r - z_{\{i,j\}} - z_{\{v_i,v_j\}} - x_{v_i,j} - x'_{v_i,j} - l_{i,j}$ (i.e., one more layer before the root r). It is easy to see that the length bound and size bound are still satisfied.

Assume there is a solution for the SLSN instance $(G', L, H_{k,k})$ with cost $4k^4 - 4k^3 + 2k^2 + 2k$. The proof that there exists a multi-colored clique of size k in G is essentially the same as Case 1, except the path between $l'_{i,j}$ and $l_{i,j}$ has one more layer.

4.3.2.3 Case 5: $\mathcal{H}_{2,k}$

In this case, we slightly modify the reduction of (Feldmann and Marx, 2016). We first change all the edges from directed to undirected. In addition, in (Feldmann and Marx, 2016) the demand graph is precisely a 2-by-$k(k-1)$ bipartite graph, but we also handle the generalization in which there may be more demands between vertices on each sides (i.e., the 2-by-$k(k-1)$ bipartite graph is just a subgraph of our demands). In order to do this, we add some dummy vertices and some edges.

Given an MCC instance $(G = (V, E), c)$ with parameter k, and a demand graph $H \in \mathcal{H}_{2,k}$, we create a unit-length and unit-cost SLSN instance G' with demand isomorphic to H as follows.

We first create a weighted graph $G_{2,k} = (V_{2,k}, E_{2,k})$. The vertex set $V_{2,k}$ contains 5 layers of vertices. The first layer V_1 is just two roots r_1, r_2. The

second layer V_2 contains a vertex $z_{\{i,j\}}$ for each $1 \leq i < j \leq k$, and a vertex y_i for each $i \in [k]$. The third layer V_3 contains a vertex z_e for each $e \in E$, and a vertex y_v for each $v \in V$. The fourth layer V_4 contains a vertex $x_{v,j}$ for each $v \in V$ and $j \neq c(v)$. The fifth layer V_5 contains a vertex $l_{i,j}$ for each $i, j \in [k]$ where $i \neq j$.

The edge set $E_{2,k}$ contains the following edges:

- $E_{11} = \{\{r_1, z_{\{i,j\}}\}, 1 \leq i < j \leq k\}$, each edge in E_{11} has length 1.
- $E_{12} = \{\{z_{\{c(u),c(v)\}}, z_e\} \mid e = \{u, v\} \in E\}$, each edge in E_{12} has length 1.
- $E_{13} = \{\{z_e, x_{u,c(v)}\} \mid e = \{u, v\} \in E\}$, each edge in E_{13} has length 1. Note that if $\{z_e, x_{u,c(v)}\} \in E_{13}$, then $\{z_e, x_{v,c(u)}\} \in E_{13}$
- $E_{21} = \{\{r_2, y_i\} \mid i \in [k]\}$, each edge in E_{21} has length 1.
- $E_{22} = \{\{y_{c(v)}, y_v\} \mid v \in V\}$, each edge in E_{22} has length 1.
- $E_{23} = \{\{y_v, x_{v,j}\} \mid v \in V, j \neq c(v)\}$, each edge in E_{23} has length 1.
- $E_{xl} = \{\{x_{v,j}, l_{c(v),j}\} \mid v \in V, j \neq c(v)\}$, each edge in E_{xl} has length 4.
- $E_{ll} = \{\{l_{i,j}, l_{i',j'}\} \mid i, j, i', j' \in [k], i \neq j, i' \neq j', (i, j) \neq (i', j')\}$, each edge in E_{ll} has length 7.

We get a unit-length graph G' from $G_{2,k}$ by replacing every edge $e \in E_{2,k}$ by a $length(e)$-hop path. Our SLSN instance consists of the graph G', length bound $L = 7$, and the following demands (which will be isomorphic to H). For each $r \in \{r_1, r_2\}$ and $i, j \in [k]$ with $i \neq j$, there is a demand between r and $l_{i,j}$ (note that these demands form a 2 by $k(k-1)$ complete bipartite

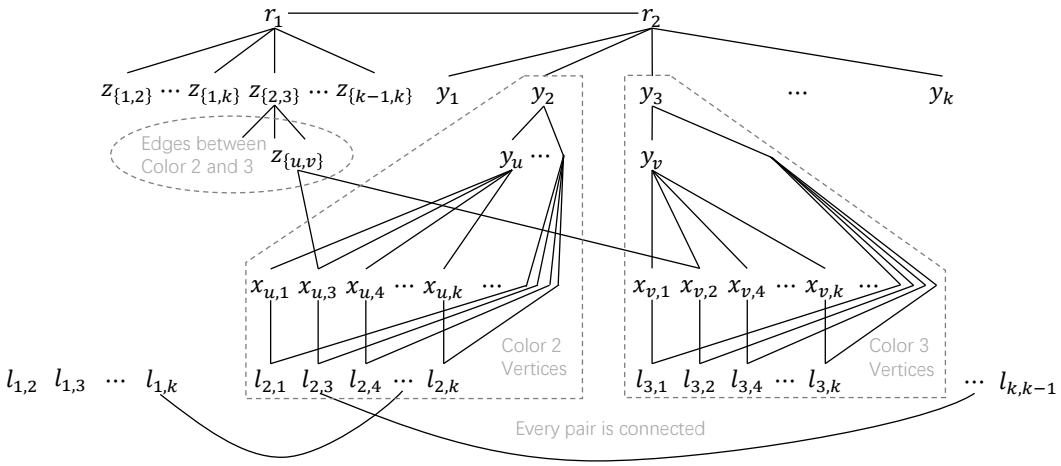

Figure 4.2: $G_{2,k}$

graph. Let this complete bipartite subgraph be B. For the rest of the demands, we arbitrarily choose a mapping between $V_1 = \{r_1, r_2\}$ and the 2-side of the bipartite graph in H, as well as a mapping between $V_5 = \{l_{i,j} \mid i, j \in [k], i \neq j\}$ and the $k(k-1)$-side. There is a demand between two vertices $u, v \in V_1 \cup V_5$ if there is an edge between u, v in H.

This construction clearly takes $poly(|V||H|)$ time. Let $g(H) = 7|H| - 7k^2 + 9k - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H}$, where $\mathbb{1}_{\{r_1,r_2\}\in H}$ is an indicator variable for $\{r_1, r_2\}$ being a demand in H. This function is also computable in time $poly(|H|)$. We first prove the easy direction in the correctness of the reduction.

Lemma 4.3.10. *If there is a multi-colored clique of size k in G, then there is a solution S for the* SLSN *instance* (G', L) *with demand graph* $H \in \mathcal{H}_{2,k}$*, and the total cost of S is* $7|H| - 7k^2 + 9k - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H}$.

Proof. Let $v_1, \ldots, v_k$ be a multi-colored clique of size k in G, where $v_i \in C_i$ for all $i \in [k]$. We create a feasible solution S to our SLSN instance, which contains following paths in G' (i.e., edges in $G_{2,k}$):

- $\{r_1, z_{\{i,j\}}\}$ for each $1 \leq i < j \leq k$. The total cost of these edges is $\binom{k}{2} = \frac{k^2-k}{2}$.

- $\{z_{\{i,j\}}, z_{\{v_i,v_j\}}\}$ for each $1 \leq i < j \leq k$. The total cost of these edges is $\binom{k}{2} = \frac{k^2-k}{2}$.

- $\{z_{\{v_i,v_j\}}, x_{v_i,j}\}$ and $\{z_{\{v_i,v_j\}}, x_{v_j,i}\}$ for each $1 \leq i < j \leq k$. The total cost of these edges is $2 \cdot \binom{k}{2} = k^2 - k$.

- $\{r_2, y_i\}$ for each $i \in [k]$. The total cost of these edges is k.

- $\{y_i, y_{v_i}\}$ for each $i \in [k]$. The total cost of these edges is k.

- $\{y_{v_i}, x_{v_i,j}\}$ for each $i, j \in [k]$ where $i \neq j$. The total cost of these edges is $2 \cdot \binom{k}{2} = k^2 - k$.

- $\{x_{v_i,j}, l_{i,j}\}$ for each $i, j \in [k]$ where $i \neq j$. The total cost of these edges is $4 \cdot 2 \cdot \binom{k}{2} = 4k^2 - 4k$.

- $\{u, v\}$ for each $\{u, v\} \in H \setminus (B \cup \{\{r_1, r_2\}\})$. The total cost of these edges is $7 \cdot (|H| - 2 \cdot k(k-1) - \mathbb{1}_{\{r_1,r_2\} \in H}) = 7|H| - 14k^2 + 14k - 7 \cdot \mathbb{1}_{\{r_1,r_2\} \in H}$.

Therefore, the total cost is $\frac{k^2-k}{2} + \frac{k^2-k}{2} + k^2 - k + k + k + k^2 - k + 4k^2 - 4k + 7|H| - 14k^2 + 14k - 7 \cdot \mathbb{1}_{\{r_1,r_2\} \in H} = 7|H| - 7k^2 + 9k - 7 \cdot \mathbb{1}_{\{r_1,r_2\} \in H}$.

Now we show the feasibility of this solution. For each $i, j \in [k]$ where $i \neq j$, the path between r_1 and $l_{i,j}$ is $r_1 - z_{\{i,j\}} - z_{\{v_i,v_j\}} - x_{v_i,j} - l_{i,j}$, and the path between r_2 and $l_{i,j}$ is $r_2 - y_i - y_{v_i} - x_{v_i,j} - l_{i,j}$. Both paths have length 7, which is within the length bound. For each $\{u, v\} \in H \setminus (B \cup \{\{r_1, r_2\}\})$, u and v have an edge with length 7, thus a path under the length bound exists. Finally, if

there exists a demand between r_1 and r_2, we can follow the path $r_1 - z_{\{1,2\}} - z_{\{v_1,v_2\}} - x_{v_1,2} - y_{v_1} - y_1 - r_2$, which has length 6. □

Now we prove the other direction.

Let S be an optimal solution for the SLSN instance (G', L) with demand graph $H^*_{k,0}$. If S has cost at most $4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k$, then there is a multi-colored clique of size k in G.

Lemma 4.3.11. *Let S be an optimal solution for the* SLSN *instance (G', L) with demand graph $H \in \mathcal{H}_{2,k}$. If S has cost at most $7|H| - 7k^2 + 9k - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H}$, then there is a multi-colored clique of size k in G.*

Proof. For each $i, j \in [k]$ where $i \neq j$, let $P_{1,i,j} \subseteq S$ be a (arbitrarily chosen) path between r_1 and $l_{i,j}$ with length at most 7, and $P_{2,i,j} \subseteq S$ be a (arbitrarily chosen) path between r_2 and $l_{i,j}$ with length at most 7. Let $\mathcal{P}_1 = \{P_{1,i,j} \mid i, j \in [k], i \neq j\}$, and $\mathcal{P}_2 = \{P_{2,i,j} \mid i, j \in [k], i \neq j\}$. As in lemma 4.3.9, we first show that some edges must be shared by multiple paths by calculating the total cost.

In order to satisfy the demand for each $\{l_{i,j}, l_{i',j'}\} \in H \setminus (B \cup \{\{r_1, r_2\}\})$, the only way is to use the edge between $l_{i,j}$ and $l_{i',j'}$ in E_{ll}. Otherwise, suppose the path has more than one edge, since the only edges incident on any $l_{i,j}$ have length either 4 or 7, the cost of two of these edges already exceeds the length bound. Thus the total cost of the edges in $S \cap E_{ll}$ is at least $7|H| - 7|B| - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H} = 7|H| - 14k^2 + 14k - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H}$.

We can see that each of the paths in $\mathcal{P}_1 \cup \mathcal{P}_2$ must have exactly one edge between every two adjacent levels, and they cannot have any other edges because of the length bound. Thus, each path $P_{1,i,j} \in \mathcal{P}_1$ must have form $r_1 -$

$z_{\{i,j\}} - z_{\{u,v\}} - x_{u,j} - l_{i,j}$ for some $\{u, v\} \in E$ with $u \in C_i$ and $v \in C_j$, and each path in $P_{2,i,j} \in \mathcal{P}_2$ must have form $r_2 - y_i - y_v - x_{v,j} - l_{i,j}$ for some $v \in C_i$.

By looking at the form of paths in $\mathcal{P}_1$, we can see that these paths are almost disjoint, except that $P_{1,i,j}$ and $P_{1,j,i}$ may share edge $\{r_1, z_{\{i,j\}}\} \in E_{11}$ and edge $\{z_{\{i,j\}}, z_e\} \in E_{12}$. Since paths in $\mathcal{P}_1$ only contain edges in $E_{11} \cup E_{12} \cup E_{13} \cup E_{xl}$, the cost of edges in $S \cap (E_{11} \cup E_{12} \cup E_{13} \cup E_{xl})$ must be at least $7 \cdot k(k-1) - \binom{k}{2} - \binom{k}{2} = 6k^2 - 6k$, even if every $P_{1,i,j}$ and $P_{1,j,i}$ do share edge $\{r_1, z_{\{i,j\}}\}$ and edge $\{z_{\{i,j\}}, z_e\}$.

We then look at the form of paths in $\mathcal{P}_2$. We can see that the first 3 hops of these paths only contain edges in $E_{21} \cup E_{22} \cup E_{23}$. In addition, these paths are all disjoint on edges in E_{23}. Moreover, in order to reach all $l_{i,j}$ from r_2 within length 7, these paths should contain all edges in E_{21} and at least k edges in E_{22}. Therefore, the total cost of edges in $S \cap (E_{21} \cup E_{22} \cup E_{23})$ should be at least $k(k-1) + k + k = k^2 + k$.

By summing up all these edges, the total cost of edges in S is already at least $7|H| - 14k^2 + 14k - 7 \cdot \mathbb{1}_{\{r_1,r_2\} \in H} + 6k^2 - 6k + k^2 + k = 7|H| - 7k^2 + 9k - 7 \cdot \mathbb{1}_{\{r_1,r_2\} \in H} = g(H)$, which means S cannot contain any edge that has not been counted before.

Therefore, S must contain exactly k edges in E_{22}, and each of these edges must have a different y_i as an endpoint. Thus these edges must have form $\{y_1, y_{v_1}\}, \ldots, \{y_k, y_{v_k}\}$, where $v_i \in C_i$ for all $i \in [k]$. We claim that $v_1, \ldots, v_k$ forms a (multicolored) clique in G.

For each $1 \leq i < j \leq k$, by looking at the form of paths in $\mathcal{P}_2$, we know that the path $P_{2,i,j}$ must be $r_2 - y_i - y_{v_i} - x_{v_i,j} - l_{i,j}$. Because of the total cost

limitation, the edge $\{x_{v_i,j}, l_{i,j}\} \in P_{2,i,j} \cap E_{xl}$ must also appear in some path in $\mathcal{P}_1$. By looking at the form of the paths in $\mathcal{P}_1$, the only possible path is $P_{1,i,j}$. Similarly, path $P_{2,j,i}$ must share edge $\{x_{v_j,i}, l_{j,i}\}$ with $P_{1,j,i}$. Again by looking at the form of the paths in $\mathcal{P}_1$, the edge in $\{z_{\{i,j\}}, z_e\} \in S \cap E_{12}$ which is shared by $P_{1,i,j}$ and $P_{1,j,i}$ must have $e = \{v_i, v_j\}$, which means $\{v_i, v_j\} \in E$.

Therefore, $v_1, \ldots, v_k$ forms a clique in G. □

4.3.2.4 Case 6: $\mathcal{H}_k$

We now want to construct an SLSN instance for a demand graph $H \in \mathcal{H}_k$ from an MCC instance $(G = (V, E), c)$ with parameter k; since all other cases have been handled, this will complete the proof of Theorem 4.3.4. By the definition of $\mathcal{H}_k$, for some $t \in [5]$ there is a graph $H^{(t)}$ of Case t that is an induced subgraph of H. We use Lemma 4.3.3 to find the graph $H^{(t)}$. Let $(G^{(t)}, L)$ be the SLSN instance obtained by applying our reduction for Case t to the MCC instance (G, c), and let the corresponding function be $g^{(t)}$.

We now want to transform the SLSN instance $(G^{(t)}, L)$ with demand graph $H^{(t)}$ into a new SLSN instance (G', L) with demand graph H, so that instance $(G^{(t)}, L, H^{(t)})$ has a solution with cost $g^{(t)}(H^{(t)})$ if and only if instance (G', L, H) has a solution with cost $g(H) = g^{(t)}(H^{(t)}) + L \cdot (|H| - |H^{(t)}|)$. If there is such a construction which runs in polynomial time, then there is a multi-colored clique of size k in G if and only if instance (G', L, H) has a solution with cost $g(H)$. This will then imply Theorem 4.3.4.

The graph G' is basically just $G^{(t)}$ with some additional vertices and edges from $H \setminus H^{(t)}$. For each vertex v in H but not in $H^{(t)}$, we add a new vertex v

to G'. For each edge $\{u, v\} \in H \setminus H^{(t)}$, we add an L-hop path between u and v to G'.

The construction still takes $poly(|V||H|)$ time, because the construction for the previous cases takes $poly(|V||H^{(t)}|)$ time and the construction for Case 6 takes $poly(|G^{(t)}||H|)$ time. Here $|H^{(t)}| \leq |H|$, and we know that $|G^{(t)}|$ is polynomial in $|V|$ and $|H^{(t)}|$.

Lemma 4.3.12. SLSN *instance* $(G^{(t)}, L, H^{(t)})$ *has a solution with cost* $g^{(t)}(H^{(t)})$ *if and only if instance* (G', L, H) *has a solution with cost* $g(H) = g^{(t)}(H^{(t)}) + L \cdot (|H| - |H^{(t)}|)$.

Proof. If instance $(G^{(t)}, L, H^{(t)})$ has a solution with cost $g^{(t)}(H^{(t)})$. Let the solution be $S^{(t)}$. For each $e = \{u, v\} \in H \setminus H^{(t)}$, let the new L-hop path between u and v in G' be P_e. Then $S^{(t)} \cup \bigcup_{e \in H \setminus H^{(t)}} P_e$ is a solution to G' with cost $g^{(t)}(H^{(t)}) + L \cdot (|H| - |H^{(t)}|)$.

If instance (G', L, H) has a solution with cost $g^{(t)}(H^{(t)}) + L \cdot (|H| - |H^{(t)}|)$, let the solution be S. Since for each $e = \{u, v\} \in H \setminus H^{(t)}$, the only path between u and v in G' within the length bound is the new L-hop path P_e, any valid solution must include all these P_e, which has total cost $L \cdot (|H| - |H^{(t)}|)$. In addition, for each demand $\{u, v\}$ which is also in H, any path between u and v in G' within the length bound will not include any new edge, because otherwise it will strictly contain an L-hop path, and have length more than L. Therefore, $S \setminus \bigcup_{e \in H \setminus H^{(t)}} P_e$ is a solution to $G^{(t)}$ with cost $g^{(t)}(H^{(t)})$. □

4.3.3 Proof of Theorem 4.1.5:

If $\mathcal{C}$ is a recursively enumerable class, and $\mathcal{C} \not\subseteq \mathcal{C}_\lambda \cup \mathcal{C}^*$ for any constant λ, then for every $k \geq 2$, let H_k be the first graph in $\mathcal{C}$ where H_k is not a star and has at least $2k^{10}$ edges. The time for finding H_k is $f(k)$ for some function f. From Lemma 4.3.3 we know that $H_k \in \mathcal{H}_k$, so that we can use Theorem 4.3.4 to construct the SLSN$_\mathcal{C}$ instance with demand H_k.

The parameter $p = |H_k|$ of the instance is a function just of k, and the construction time is FPT from Theorem 4.3.4. Therefore this is a FPT reduction from the MCC problem to the unit-length unit-cost SLSN$_\mathcal{C}$ problem. Thus Theorem 4.3.2 implies that the unit-length unit-cost SLSN$_\mathcal{C}$ problem is W[1]-hard for parameter p. □

4.4 Algorithms for the Arbitrary-Length Arbitrary-Cost SLSN

The idea for the algorithms for arbitrary-length arbitrary-cost SLSN is the same as that for unit-length arbitrary cost. However, the arbitrary-lengths increase the difficulty of the problem. For example, we cannot use Bellman-Ford algorithm to find the lowest cost path within a certain distance bound. We also cannot go over all the possible lengths of the paths in the dynamic programming algorithm, because it will take exponential time.

In order to recover from this, we utilize some techniques in the $(1+\varepsilon)$-approximation algorithm for the RESTRICTED SHORTEST PATH problem, where the problem is the special case of SLSN with $p = 1$.

4.4.1 Preliminaries

In this section, we will introduce two algorithms: the first one gives a bound on the optimal solution of the SLSN problem, and the second one finds the shortest path under some flexible cost constraint. The algorithms are similar to the algorithms for the RESTRICTED SHORTEST PATH problem.

Given an SLSN instance, the first algorithm *OptLow* orders all the edges in E by cost and starts from the lowest one. In each iteration i, let e_i be the edge considered, and let G_{e_i} be the graph that contains all edges in E with cost at most $c(e_i)$. The algorithm checks if G_{e_i} contains a feasible solution of the SLSN instance, and returns $C = c(e_i)$ if a solution exists. The pseudocode is in Algorithm 5.

Algorithm 5 $OptLow(G = (V, E), c, l, H)$

```
Order all the edges in E by the cost and get c(e_1) ≤ c(e_2) ≤ ... ≤ c(e_|E|)
for i = 1, ..., |E| do
    G_{e_i} ← the graph that contains all edges in E with cost at most c(e_i)
    if G_{e_i} has a feasible solution then
        C ← c(e_i)
        break
    end if
end for
return C
```

Lemma 4.4.1. *Let C be the solution returned by Algorithm 5 and OPT be the cost of the optimal solution for the* SLSN *instance $(G = (V, E), c, l, H)$. Then $C \leq OPT \leq n^2 C$, and the running time is polynomial in n.*

Proof. Since a graph in which all edges have cost less than C does not have a feasible solution, we know that every feasible solution contains an edge which

has cost at least C, so the optimal solution has cost at least C.

For the upper bound, because graph G_{e_i} contains a feasible subgraph, so there is a feasible solution with cost at most n^2C, thus the optimal solution has cost at most n^2C.

Because we can use a standard shortest path algorithm for each pair of demands to test the feasibility, we can see that the running time is polynomial in n. □

Following is the second algorithm, which is aiming to find a low cost path under certain distance bound. Algorithm 6 is essentially a dynamic programming algorithm on graph G, which first defined new costs $c_e^* = \left\lceil \frac{n \cdot c(e)}{\varepsilon C} \right\rceil$ for all $e \in E$, and then calculate the shortest path from s to t with bounded new cost $\lfloor \frac{n}{\varepsilon} \rfloor$.

Lemma 4.4.2. *Given a graph $G = (V, E)$, a cost function c, a length function l, a pair of vertices (s, t), a constant $\varepsilon > 0$, and a cost bound C, if there exists a path between s and t with cost at most $(1 - 2\varepsilon)C$ and distance D, then Algorithm 6 returns a path between s and t with cost at most C and distance at most D. The running time is polynomial in $\frac{n}{\varepsilon}$.*

Proof. The running time of this algorithm is clearly polynomial in $\frac{n}{\varepsilon}$ because there is only $poly(\frac{n}{\varepsilon})$ slots of $d(v, i)$.

We first claim that $d(v, i)$ is the minimal length of a path from s to v under new cost i. This can be proven easily by induction. The base case is $i = 0$, where $d(s, 0) = 0$, and for other $v \in V \setminus \{s\}$, the minimal length is infinity. For the inductive step, we consider the last edge $\{u, v\}$ of the shortest path from s

Algorithm 6 $MinDist(G = (V, E), c, l, s, t, \varepsilon, C)$

for $e \in E$ **do**
  $c_e^* \leftarrow \left\lceil \frac{n \cdot c(e)}{\varepsilon C} \right\rceil$
end for
for $v \in V \setminus \{s\}$ **do**
  $d(v, 0) \leftarrow \infty$
end for
$d(s, 0) \leftarrow 0$
for $i = 1, \ldots, \left\lfloor \frac{n}{\varepsilon} \right\rfloor$ **do**
  for $v \in V$ **do**
    $d(v, i) \leftarrow \min_{e=(u,v) \in E, c_e^* \leq i} l(e) + d(u, i - c_e^*)$
  end for
end for
$C^* \leftarrow \arg\min_{i \in [\left\lfloor \frac{n}{\varepsilon} \right\rfloor]} d(t, i)$
if $d(t, C^*) < \infty$ **then**
  return the corresponding path for $d(t, C^*)$
else
  return $\varnothing$
end if

to v under new cost i. By removing this edge from the path, it must be a path from s to u under new cost $i - c^*_{\{u,v\}}$. Therefore our claim holds.

If there exists a path between s and t with cost at most $(1 - 2\varepsilon)C$ and distance D, let the edge set of this path be S. Then because S has at most n edges, we know that the new total cost of S is

$$\sum_{e \in S} c_e^* = \sum_{e \in S} \left\lceil \frac{n \cdot c(e)}{\varepsilon C} \right\rceil \leq \frac{n \cdot \sum_{e \in S} c(e)}{\varepsilon C} + n \leq \frac{n(1 - 2\varepsilon)C}{\varepsilon C} + n \leq \left\lfloor \frac{n}{\varepsilon} \right\rfloor.$$

Thus the algorithm must return a non-empty set S', which connects s and t with distance at most D.

Now we can calculate the original cost of path S'. We know that

$$\left\lfloor \frac{n}{\varepsilon} \right\rfloor \geq \sum_{e \in S'} c_e^* = \sum_{e \in S'} \left\lceil \frac{n \cdot c(e)}{\varepsilon C} \right\rceil > \frac{n \cdot \sum_{e \in S'} c(e)}{\varepsilon C}.$$

Thus

$$\sum_{e \in S'} c(e) < \left\lfloor \frac{n}{\varepsilon} \right\rfloor \cdot \frac{\varepsilon C}{n} \leq \frac{n}{\varepsilon} \cdot \frac{\varepsilon C}{n} = C.$$

Therefore, the algorithm returns a path S', which has cost at most C and distance at most D. □

4.4.2 Constant Number of Demands

With the algorithms in Section 4.4.1, we can now introduce our FPTAS algorithm for arbitrary-length arbitrary-cost $\text{SLSN}_{\mathcal{C}_\lambda}$. Since Lemma 4.2.1 still holds for the arbitrary length case, we can use the same idea as in Algorithm 4. However, in the arbitrary length case we can not guess the exact length of each subpath, because there are too many possible lengths. We even can not guess an approximate length for each subpath, because any violation on the length bound may make the solution infeasible. Therefore, we switch to guess the approximate cost of each subpath to solve this problem.

The algorithm for arbitrary-length arbitrary-cost $\text{SLSN}_{\mathcal{C}_\lambda}$ first bound the optimal cost using Algorithm 5. Then guess the endpoint set Q and the set E' which intuitively represents how the vertices in $Q \cup \bigcup_{i=1}^{p} \{s_i, t_i\}$ connected to each other, the same way as in Algorithm 4. After that, the algorithm switch to guess the cost c' of each subpath, basically up to a $(1+\varepsilon)$ error. Finally, the algorithm connect each pair of $u, v \in V$ where $\{u, v\} \in E'$ by shortest paths with restricted cost $c'(\{u, v\})$ using Algorithm 6, check the feasibility, and

output the optimal solution. The detailed algorithm is in Algorithm 7.

Algorithm 7 Arbitrary-length arbitrary-cost $\text{SLSN}_{\mathcal{C}_\lambda}$

$C \leftarrow OptLow(G = (V, E), c, l, H)$
$M \leftarrow \sum_{e \in E} c(e)$
$S \leftarrow E$
for $Q \subseteq V$ where $|Q| \leq p(p-1)$ **do**
  $Q' \leftarrow Q \cup \bigcup_{i=1}^{p} \{s_i, t_i\}$
  for $E' \subseteq \{\{u, v\} \mid u, v \in Q', u \neq v\}$, $c' : E' \to \{-\lceil 2\log_{1+\varepsilon} n \rceil, \ldots, -1, 0, 1, \ldots, \lceil 2\log_{1+\varepsilon} n \rceil + 3\}$ **do**
    $T \leftarrow \varnothing$
    for $\{u, v\} \in E'$ **do**
      $T \leftarrow T \cup MinDist(G, c, l, u, v, \varepsilon, (1+\varepsilon)^{c'(\{u,v\})} C)$
    end for
    if T is a feasible solution and $\sum_{e \in T} c(e) < M$ **then**
      $M \leftarrow \sum_{e \in T} c(e)$
      $S \leftarrow T$
    end if
  end for
end for
return S

Claim 4.4.3. *The running time for Algorithm 7 is is* $(\frac{n}{\varepsilon})^{O(p^4)}$.

Proof. We know that $OptLow$ can be done in polynomial time by Lemma 4.4.1. Similar to Algorithm 4, we can also see that there are at most $n^{p(p-1)}$ possible Q, and for each Q there are at most $2^{(p(p-1)+2p)^2}$ possible E', and at most $(2 \cdot \lceil 2\log_{1+\varepsilon} n \rceil + 3)^{(p(p-1)+2p)^2}$ possible c'. The algorithm 6 in the inner loop takes $poly(\frac{n}{\varepsilon})$ running time. Thus the total running time is at most $n^{p(p-1)} \cdot 2^{(p(p+1))^2} \cdot (\frac{5\log n}{\varepsilon})^{(p(p+1))^2} \cdot poly(\frac{n}{\varepsilon}) + poly(n)$. □

4.4.2.1 Proof of Theorem 4.1.6:

The running time has been proven in Claim 4.4.3, which is polynomial in $\frac{n}{\varepsilon}$ if $\lambda \geq p$ is a constant. The correctness is also similar to Algorithm 4. Because the algorithm only returns feasible solution, so we only need to show that this algorithm returns a solution with cost at most the cost of the optimal solution.

We define S^*, P_i^*, Q^*, Q'^*, l'^*, $P^*_{\{u,v\}}$ the same as in the proof of Theorem 4.1.3. We further define $c'^*(\{u,v\})$ as the cost of $P^*_{\{u,v\}}$ for each $\{u,v\} \in E'$.

Since the algorithm iterates over all possibilities for Q, E' and c', there is some iteration in which $Q = Q'^*$, $E' = E'^*$, and

$$c' \equiv \max\left\{\left\lceil \log_{1+\varepsilon} \frac{c'^*}{(1-2\varepsilon)C}\right\rceil, -\left\lceil 2\log_{1+\varepsilon}\frac{n^2}{\varepsilon}\right\rceil\right\}.$$

The reason that this c' must have been iterated is because of Lemma 4.4.1. We can see that, $c'^*(\{u,v\}) \leq OPT \leq n^2C$ for every $\{u,v\} \in E'^*$, and so that

$$\left\lceil \log_{1+\varepsilon} \frac{c'^*(\{u,v\})}{(1-2\varepsilon)C}\right\rceil \leq \left\lceil \log_{1+\varepsilon} \frac{n^2C}{(1-2\varepsilon)C}\right\rceil \leq \lceil 2\log_{1+\varepsilon} n\rceil + 3.$$

We will show that the algorithm also must find a feasible solution in this iteration.

For each $i \in [p]$, the path P_i^* is partitioned to edge-disjoint subpaths by Q'^*. Let q_i be the number of subpaths, and let the endpoints be $s_i = v_{i,0}, v_{i,1}, \ldots, v_{i,q_i-1}, v_{i,q_i} = t_i$. We further let these subpaths be

$$P^*_{\{s_i,v_{i,1}\}}, P^*_{\{v_{i,1},v_{i,2}\}}, \ldots, P^*_{\{v_{i,q_i-1},t_i\}}.$$

By the definition of l'^* and c'^*, for each $j \in [q_i]$, there must be a path between

$v_{i,j-1}$ and $v_{i,j}$ with length at most $l'^*(\{v_{i,j-1}, v_{i,j}\})$ and cost at most

$$c'^*(\{v_{i,j-1}, v_{i,j}\}) \leq (1-2\varepsilon) \cdot (1+\varepsilon)^{\{c'(v_{i,j-1}, v_{i,j}\})} C.$$

Therefore, by Lemma 4.4.2 we know that the edge set T in this iteration must contains a path between u and v with length at most $l'^*(\{v_{i,j-1}, v_{i,j}\})$ and cost at most

$$(1+\varepsilon)^{c'(\{v_{i,j-1}, v_{i,j}\})} C \leq \max\left\{\frac{c'^*(\{v_{i,j-1}, v_{i,j}\})}{1-2\varepsilon}, \frac{\varepsilon C}{n^2}\right\}.$$

Because the summation $\sum_{j=1}^{q_i} l'^*(\{v_{i,j-1}, v_{i,j}\})$ is at most L, we know that the edge set T in this iteration must satisfies demand $\{s_i, t_i\}$ for each $i \in [p]$. Therefore it is a feasible solution.

We can also see that the cost of the edge set T in this iteration is at most

$$\begin{aligned}
\sum_{\{u,v\} \in E'^*} \max\left\{\frac{c'^*(\{u,v\})}{1-2\varepsilon}, \frac{\varepsilon C}{n^2}\right\} &\leq \frac{1}{1-2\varepsilon} \sum_{e \in E'^*} c'^*(\{u,v\}) + \varepsilon C \\
&\leq \frac{OPT}{1-2\varepsilon} + \varepsilon C \qquad (4.2) \\
&\leq \frac{OPT}{1-2\varepsilon} + \varepsilon OPT \qquad (4.3) \\
&\leq (1+4\varepsilon) OPT.
\end{aligned}$$

Equation (4.2) is because the all the $P^*_{u,v}$ are edge-disjoint, and thus we have $OPT = \sum_{(u,v) \in E'^*} c'^*(\{u,v\})$. Equation (4.3) is because we know that $OPT \geq C$ by Lemma 4.4.1.

Therefore, the algorithm outputs a $(1+4\varepsilon)$-approximation of the optimal

solution. By replacing ε with $\frac{\varepsilon}{4}$ in the whole algorithm, we get a $(1+\varepsilon)$-approximation. □

4.4.3 Star Demand Graphs ($\mathbf{SLSN}_{\mathcal{C}^*}$)

For the case that the demand graph is a star, let $s = s_1 = s_2 = \ldots = s_p$, and $T = \{t_1, \ldots, t_p\}$.

We first bound the optimal cost using Algorithm 5, and then assign a new cost for each edge depending our bound of the optimal cost. Finally, we use a dynamic programming algorithm which is similar to the algorithm for DST to solve the problem under the new edge costs, and we can show that it is a $(1+\varepsilon)$-approximation to the optimal solution in the original edge cost. The detailed algorithm is in Algorithm 8.

Here we are aiming to set $d(v, R, j)$ as the smallest height of a tree, such that the root is v, the total new cost is at most j, and it contains all the vertices in R. Then, we can find the minimal j which makes $d(v, T, j) \leq L$, and this j is the minimal cost of a feasible solution under the new cost. Note that we only need to consider the height of trees because the optimal solution in this case is always a tree.

Lemma 4.4.4. *The optimal solution S^* of the* $\mathrm{SLSN}_{\mathcal{C}^*}$ *problem is always a tree.*

Proof. We can assign path $P_1, \ldots P_p$ as in the Lemma 4.2.1. Because S^* is an optimal solution, it will not contain any edge other than the edges in $P_1, \ldots P_p$. If there is a cycle in S^*, then there are two paths P_i and P_j intersect at a vertex v other than the root s, and the paths to v are different, which contradict with Lemma 4.2.1. Therefore S^* is always a tree. □

Algorithm 8 Arbitrary-length arbitrary-cost $\text{SLSN}_{\mathcal{C}^*}$

$C \leftarrow OptLow(G = (V, E), c, l, H)$
for $e \in E$ **do**
  $c_e^* \leftarrow \left\lceil \frac{n \cdot c(e)}{\varepsilon C} \right\rceil$
end for
for $v \in V, R \subseteq T$, and $j \in [\left\lceil \frac{n^3(1+\varepsilon)}{\varepsilon} \right\rceil]$ **do**
  $d(v, R, j) \leftarrow \begin{cases} 0, & \text{if } |R| = 1 \text{ and } v \in R, \text{ or } R = \varnothing \\ \infty, & \text{otherwise} \end{cases}$
end for
for $j = 1, \ldots, \left\lceil \frac{n^3(1+\varepsilon)}{\varepsilon} \right\rceil$ **do**
  for $i = 1, \ldots, p$ **do**
    for $v \in V, R \subseteq T$ with $|R| = i$ **do**
      $d(v, R, j) \leftarrow \min_{v' \in V, R' \subseteq R, k \leq j - c^*_{\{v,v'\}}} (l(\{v, v'\}) + \max\{d(v', R' \setminus \{v\}, k), d(v', R \setminus R' \setminus \{v\}, j - c^*_{\{v,v'\}} - k)\})$
    end for
  end for
end for
for $j = 1, \ldots, \left\lceil \frac{n^3(1+\varepsilon)}{\varepsilon} \right\rceil$ **do**
  if $d(s, T, j) \leq L$ **then**
    return the corresponding tree for $d(s, T, j)$
  end if
end for

We again first prove the running time.

Claim 4.4.5. *Algorithm 8 runs in time* $O(4^p \cdot poly(\frac{n}{\varepsilon}))$.

Proof. Because running algorithm $OptLow$ and setting new costs runs in polynomial time, we only need to prove the time of the dynamic programming part. We can see that, d has at most $n \cdot 2^p \cdot \left\lceil \frac{n^3(1+\varepsilon)}{\varepsilon} \right\rceil$ slots, and filling each of them takes at most $n \cdot 2^p$ time, so the total running time is $O(4^p \cdot poly(\frac{n}{\varepsilon}))$. □

We then prove the correctness of the dynamic programming part.

Lemma 4.4.6. *For each* $v \in V$, $S \subseteq T$, *and* $j \in [\left\lceil \frac{n^3(1+\varepsilon)}{\varepsilon} \right\rceil]$, *the* $d(v, R, j)$ *stores the smallest height of a tree, such that the root is* v, *the total new cost is at most* j, *and it contains all the vertices in* R.

Proof. We prove the lemma using induction. The base case is that $R = \varnothing$ or $R = \{v\}$, which has already been initialized.

The algorithm fill all the $d(v, R, j)$ with the ascending order of j and then ascending order of $|R|$, so for any $v' \in V$, $R' \subseteq R$, and $k \leq j - c^*_{\{v,v'\}}$, we know that $d(v', R' \setminus \{v\}, k)$ and $d(v', R \setminus R' \setminus \{v\}, j - c^*_{\{v,v'\}} - k)$ must have already been filled before filling $d(v, R, j)$. We will show that when updated, $d(v, R, j)$ is at most and at least the smallest height of a tree, such that the root is v, the total new cost is at most j, and it contains all the vertices in R.

For the "at most" part, let S be the lowest tree, such that the root is v, the total new cost is at most j, and it contains all the vertices in R. If S is not in the base case, then either v has degree 1, or v has degree more than 1 in S.

If v has degree 1, then there must be a v' which is adjacent to v, and the tree rooted at v' is the lowest height tree, which contains all the vertices in $R \setminus \{v\}$,

and the total cost is at most $j - c^*_{\{v,v'\}}$. This case is already considered in the algorithm by setting $R' = R$ and $k = j - c^*_{\{v,v'\}}$, thus in this case $d(v, R, j)$ is at most the height of S.

If v has degree more than 1, then S can be split to two trees S_1 and S_2 with the same root v. Let $R_1 = R \cap S_1$, then the height of S is at least the lowest possible height of S_1, and also at least the lowest possible height of S_2. This case is considered in the algorithm by setting $v' = v$, $R' = R_1$ and k be the cost of S_1, thus in this case $d(v, R, j)$ is also at most the height of S.

For the "at least" part, we only need to show that there exist a tree with height $d(v, R, j)$ such that the root is v, the total new cost is at most j, and it contains all the vertices in R. Let v^*, R^*, and k^* be the value of v', R, and k which gives the minimum value of $d(v, R, j)$. Then after removed redundant edges, the union of the edge $\{u, v\}$, the tree for $d(v^*, R^* \setminus \{v\}, k^*)$, and the tree for $d(v^*, R \setminus R^* \setminus \{v\}, j - c^*_{\{v,v^*\}} - k^*)$ is a tree which the root is v, the total new cost is at most j, and it contains all the vertices in S, with height $d(v, R, j)$.

Therefore $d(v, R, j)$ is correctly set to what we want. □

No we can finally prove our Theorem 4.1.7.

4.4.3.1 Proof of Theorem 4.1.7

The running time has already been proven in Claim 4.4.5. Now we prove the correctness.

Let S^* be the optimal solution and let $OPT = \sum_{e \in S^*} c(e)$. Then, the new

cost of this solution is at most

$$\begin{aligned}\sum_{e\in S^*} c_e^* &= \sum_{e\in S^*} \left\lceil \frac{n\cdot c(e)}{\varepsilon C} \right\rceil \\ &\leq \frac{n\cdot \sum_{e\in S^*} c(e)}{\varepsilon C} + n \\ &\leq \frac{n\cdot OPT}{\varepsilon C} + \frac{n\cdot \varepsilon OPT}{\varepsilon C} \\ &= \frac{n}{\varepsilon C}(1+\epsilon)OPT \\ &\leq \left\lceil \frac{n^3(1+\varepsilon)}{\varepsilon} \right\rceil,\end{aligned}$$

because from Lemma 4.4.4 we know that $|S^*| \leq n$ and from Lemma 4.4.1 we know that $C \leq OPT \leq n^2 C$.

Since S^* is a tree with height at most L, such that the root is s, the total new cost is at most $\frac{n}{\varepsilon C}(1+\epsilon)OPT$, and it contains all the vertices in T, from Lemma 4.4.6 and the last section of Algorithm 8 we know that the algorithm must returns a tree S with height at most L, the total new cost is at most $\frac{n}{\varepsilon C}(1+\epsilon)OPT$, and it contains all the vertices in T, which is a feasible solution.

Now we calculate the original cost of S. Because

$$\frac{n}{\varepsilon C}(1+\epsilon)OPT \geq \sum_{e\in S} c_e^* = \sum_{e\in S} \left\lceil \frac{n\cdot c(e)}{\varepsilon C} \right\rceil \geq \frac{n\cdot \sum_{e\in S} c(e)}{\varepsilon C},$$

we know that $\sum_{e\in S} c(e) \leq (1+\epsilon)OPT$, which is a $(1+\varepsilon)$ approximation to the optimal solution. □

4.5 Hardness for the Unit-Length Polynomial-Cost SLSN

4.5.1 Preliminaries

Here, we will do a FPT reduction from the MULTI-COLORED DENSEST k-SUBGRAPH (MULTI-COLORED DkS) problem to the unit-length polynomial-cost SLSN$_{\mathcal{C}}$ problem with a $\mathcal{C} \not\subseteq \mathcal{C}_\lambda \cup \mathcal{C}^*$. Here is the definition of the MULTI-COLORED DkS problem.

Definition 4.5.1 (MULTI-COLORED DENSEST k-SUBGRAPH)**.** Given a graph $G = (V, E)$, a number $k \in \mathbb{N}$, a coloring function $c : V \to [k]$, and a factor $\alpha < 1$. The objective of the MULTI-COLORED DkS problem is to distinguish the following two cases:

- There is a k-clique in G, where each vertex has different color.
- Every subgraph of G induced by k vertices contains less than $\alpha \cdot \binom{k}{2}$ edges.

Previously, it has been proven that, assume Gap-ETH holds, then there is no FPT algorithm for MULTI-COLORED DkS even with $\alpha = o(1)$. Formally, the theorem is as follows.

Theorem 4.5.2 (Chitnis, Feldmann, and Manurangsi, 2017, Corollary 24)**.** *Assuming (randomized) Gap-ETH, for any function $h(k) = o(1)$ and any function f, there is no $f(k) \cdot n^{O(1)}$-time algorithm that solves* MULTI-COLORED DkS *with factor $\alpha = k^{-h(k)}$.*

We can easily get a weaker version of this theorem which $\alpha = O(1)$.

Corollary 4.5.3. *For any constant $0 < \alpha < 1$, for any function f, assuming (randomized) Gap-ETH there is no $f(k) \cdot n^{O(1)}$-time algorithm that solves* Multi-Colored DkS *with factor α.*

Proof. We can set $h(k) = \log_k \frac{1}{\alpha}$ in Theorem 4.5.2. □

4.5.2 Reduction

Theorem 4.5.4. *Let $1 \geq \epsilon > 0$ be an arbitrary constant, and let $G = (V, E)$, coloring function $c : V \to [k]$, and factor ε be a* Multi-Colored DkS *instance. Let $H \in \mathcal{H}_k$. Then we can construct a unit-length polynomial-cost* SLSN *instance (G', L) with demand graph H in $poly(|V||H|)$ time, and there exists a function g (computable in time $poly(|H|)$) such that*

- *If there is a k-clique in G, where each vertex has different color, then the* SLSN *instance has a solution with cost $g(H)$.*
- *If every subgraph of G induced by k vertices contains less than $\varepsilon \cdot \binom{k}{2}$ edges, then the optimal cost of the* SLSN *instance is at least $\left(\frac{5}{4} - \varepsilon\right) g(H)$.*

As in the unit-length unit-cost setting, we will first design a reduction for demand graphs $H \in \{H^*_{k,0}, H^*_{k,1}, H^*_{k,2}, H_{k,k}\} \cup \mathcal{H}_{2,k}$ first, and then consider the general $H \in \mathcal{H}_k$.

4.5.2.1 Case 1: $H^*_{k,0}$

Let $G = (V, E)$ with coloring function $c : V \to [k]$ and factor ε be a Multi-Colored DkS instance. We create a unit-length and polynomial-cost SLSN instance G' with demand graph $H^*_{k,0}$ as following.

We again use the length-weighted graph G_k^* constructed in Section 4.3.2.1. We change the cost of edges in $E_2 \cup E_4$ to $4k^4$, while keeping the cost equal to the length for the rest of the edges.

G' is again a graph that each edge $e \in E_k^*$ is replaced by a $length(e)$-hop path. Where the cost of edges is divided equally for each hop. The demands are the same as the demands in Section 4.3.2.1, and L is still $4k^2$. The construction still takes $|V||H_{k,0}^*|$. The function g is slightly different, where $g(H_{k,0}^*) = 6k^6 - 6k^5 + 3k^4 + k$, this function is also computable in $poly(H_{k,0}^*)$ time.

Using the same solution as in the proof of Lemma 4.3.5, we can see that, If there is a multi-colored clique of size k, then the SLSN instance has a solution with cost

$$4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k + \left(\binom{k}{2} + k(k-1)\right) \cdot (4k^4 - 1) = 6k^6 - 6k^5 + 3k^4 + k,$$

because the cost of $\binom{k}{2} + k(k-1)$ edges in this solution is changed from 1 to $4k^4$.

The other direction for the correctness is the following lemma.

Lemma 4.5.5. *Let S be an optimal solution for the* SLSN *instance (G', L) with demand graph $H_{k,0}^*$. If S has cost at most $\left(\frac{5}{4} - \frac{\varepsilon}{4}\right) g(H_{k,0}^*)$, then there is a subgraph of G with $\varepsilon \cdot \binom{k}{2}$ edges.*

Proof. For each $i, j \in [k]$ where $i \neq j$, let $P_{i,j}$ be a (arbitrarily chosen) path in S which connects r and $l_{i,j}$ with length at most $L = 4k^2$. Let $\mathcal{P} = \{P_{i,j} \mid i, j \in [k], i \neq j\}$ be the set of all these paths. We also let P_y be a (arbitrarily chosen) path in S which connects y_0 and y_k with length at most L.

From Claim 4.3.8, P_y can be divided to k subpaths, each correlates to a vertex v_i. We will show that the induced subgraph on vertex set $\{v_1, \ldots, v_k\}$ has at least $\varepsilon \cdot \binom{k}{2}$ edges. In fact, let $R = \{\{v_i, v_j\} \mid i, j \in [k], i \neq j, P_{i,j} \cap E_2 = P_{j,i} \cap E_2, P_y \cap P_{i,j} \cap E_4 \neq \varnothing, P_y \cap P_{j,i} \cap E_4 \neq \varnothing\}$, we will show that $R \subseteq E$ and $|R| \geq \varepsilon \cdot \binom{k}{2}$.

For any $\{v_i, v_j\} \in R$, by looking at the definition of G_k^* and the form of the path P_y and $P_{i,j}$ in Claim 4.3.6 and 4.3.8, we know that if $P_y \cap P_{i,j} \cap E_4$ is not an empty set, then it must contain only one edge $\{x_{v_i,j}, x'_{v_i,j}\}$. Similarly, $P_y \cap P_{j,i} \cap E_4$ must be the edge $\{x_{v_j,i}, x'_{v_j,i}\}$. From this, we can see that $P_{i,j} \cap E_2 = P_{j,i} \cap E_2$ must be the edge $\{z_{\{i,j\}}, z_{\{v_i,v_j\}}\}$, because $z_{\{v_i,v_j\}}$ is the only vertex which is adjacent to both $x_{v_i,j}$ and $x_{v_j,i}$. Therefore by the definition of E_2, we know that $\{v_i, v_j\}$ is an edge of E, which means $R \subseteq E$. Thus the only thing left is to show that $|R| \geq \varepsilon \cdot \binom{k}{2}$.

From Claim 4.3.8, because P_y contains k subpaths, and each subpath contains $k-1$ edges in E_4, we know that $|P_y \cap E_4| \geq k(k-1)$. From Claim 4.3.6, because for each $i, j \in [k]$ where $i \neq j$, there is at least one edge in $P_{i,j} \cap E_4$, and they must be different from each other, we know that $\left|\bigcup_{i,j\in[k],i\neq j} P_{i,j} \cap E_4\right| \geq k(k-1)$. Let $x = |S \cap E_4| = \left|(P_y \cup \bigcup_{i,j\in[k],i\neq j} P_{i,j}) \cap E_4\right|$, then

$$\left|P_y \cap \bigcup_{i,j\in[k],i\neq j} P_{i,j} \cap E_4\right| = \left|P_y \cap E_4\right| + \left|\bigcup_{i,j\in[k],i\neq j} P_{i,j} \cap E_4\right| - x \geq 2k(k-1) - x.$$

Let $T = \{P_{i,j} \mid i, j \in [k], i \neq j, P_y \cap P_{i,j} \cap E_4 \neq \varnothing\}$, then $|T| \geq 2k(k-1) - x$, because each $P_{i,j}$ can share at most one edge with P_y.

We also know that each edge $\{z_{\{i,j\}}, z_e\} \in S \cap E_2$ can only appear in at most two different paths, which are $P_{i,j}$ and $P_{j,i}$. Let $y = |S \cap E_2|$. From Claim

4.3.6 we know that each path in T must contain at least one edge in $S \cap E_2$, thus there are at least $|T| - y$ edges in $S \cap E_2$ which appear in two different paths in T. Therefore $|R| \geq |T| - y$.

Now we calculate the size of R. Because every edge in $E_2 \cup E_4$ has cost $4k^4$, and the total cost is less than $\left(\frac{5}{4} - \frac{\varepsilon}{4}\right) g(H^*_{k,0})$, thus we have

$$x + y = |S \cap E_2| + |S \cap E_4| \leq \left\lfloor \frac{\left(\frac{5}{4} - \frac{\varepsilon}{4}\right) g(H^*_{k,0})}{4k^4} \right\rfloor \leq \left(\frac{15}{8} - \frac{3\varepsilon}{8}\right) k(k-1),$$

so that

$$|R| \geq |T| - y \geq 2k(k-1) - x - y \geq 2k(k-1) - \left(\frac{15}{8} - \frac{3\varepsilon}{8}\right) k(k-1) \geq \varepsilon \cdot \binom{k}{2}.$$

Therefore the induced subgraph with vertex set $\{v_1, \ldots, v_k\}$ has at least $\varepsilon \cdot \binom{k}{2}$ edges. □

4.5.2.2 Case 2, 3, and 4:

In this setting, the construction of Case 2, 3 still keep the same as Case 1, and the change for 4 are basically the same as the unit-cost setting.

Case 2: $H^*_{k,1}$

We use the same G^*_k, G' and L in the construction of the SLSN instance for demand graph $H^*_{k,0}$, and also set $g(H^*_{k,1}) = 6k^6 - 6k^5 + 4k^4 + k$. The only difference is the demand graph. Besides the demand of $\{r, l_{i,j}\}$ for all $i, j \in [k]$ where $i \neq j$, and $\{y_0, y_k\}$, there is a new demand $\{r, y_0\}$. Clearly this new demand graph is a star with $(k(k-1)+1)$ leaves, and an edge in which exactly one of the endpoints is a leaf of the star, so it is isomorphic to $H^*_{k,1}$.

Assume there is a multi-colored clique of size k in G. The paths connecting previous demands in the solution of the SLSN instance are the same as Case 1. The path between r and y_0 is $r - z_{\{1,2\}} - z_{\{v_1,v_2\}} - x_{v_1,2} - y_0$. All the edges in this path is already in the previous paths, so the cost remains the same. The length of this path is $2 + 1 + 2k^2 - 2 + 4 = 2k^2 + 5 < 4k^2$, which satisfies the length bound.

Assume there is a solution for the SLSN instance $(G', L, H^*_{k,1})$ with total cost less than $\left(\frac{5}{4} - \frac{\varepsilon}{4}\right) g(H^*_{k,1})$. The proof of existing a subgraph of G with $\varepsilon \cdot \binom{k}{2}$ edges is the same as Case 1.

Case 3: $H^*_{k,2}$

As in Case 2, only the demand graph changes. The new demand graph is the same as in Case 2 but again with a new demand $\{r, y_k\}$. Since $\{r, y_0\}$was already a demand, our new demand graph is a star with $(k(k-1)+2)$ leaves (the $l_{i,j}$'s and y_0 and y_k), and an edge between two of its leaves (y_0 and y_k), which is isomorphic to $H^*_{k,2}$.

Assume there is a multi-colored clique of size k in G. The paths connecting previous demands in the solution of the SLSN instance are the same as Case 2. The path between r and y_k is $r - z_{\{k-1,k\}} - z_{\{v_{k-1},v_k\}} - x_{v_k,k-1} - y_k$. All the edges in this path is already in the previous paths, so the cost stays the same. The length of this path is $2 + 1 + 2k^2 - 2 + 4 = 2k^2 + 5 < 4k^2$, which satisfies the length bound.

Assume there is a solution for the SLSN instance $(G', L, H^*_{k,2})$ with total cost less than $\left(\frac{5}{4} - \frac{\varepsilon}{4}\right) g(H^*_{k,2})$. The proof of existing a subgraph of G with

$\varepsilon \cdot \binom{k}{2}$ edges is the same as Case 1.

Case 4: $H_{k,k}$

In order to get $H_{k,k}$ as our demand graph, we have to slightly change the construction in Case 1. We still first make a weighted graph $G_{k,k} = (V_{k,k}, E_{k,k})$ and then transform it to the unit-length graph G'. For the vertex set $V_{k,k}$, we add another layer of vertices $V_0 = \{l'_{i,j} \mid i, j \in [k], i \neq j\}$ in to V_k^* before the first layer V_1. For the edge set $E_{k,k}$, we include all the edges in E_k^*, but change the edges in E_1 to length 1 and cost 1. We also add another edge set $E_0 = \{\{l'_{i,j}, r\} \mid i, j \in [k], i \neq j\}$. Each edge in E_0 has length 1 and cost 1.

The demands are $\{l'_{i,j}, l_{i,j}\}$ for each $i, j \in [k]$ where $i \neq j$, as well as $\{y_0, y_k\}$. This is a matching of size $k(k-1)+1$, which is isomorphic to $H_{k,k}$. We still set the length bound to be $L = 4k^2$, and set $g(H_{k,k}) = 6k^6 - 6k^5 + 3k^4 + \frac{k^2}{2} + \frac{k}{2}$.

If there is a multi-colored clique of size k in G, the construction for the solution in G' is similar to Case 1. For each $i, j \in [k]$ where $i \neq j$, the paths between $l'_{i,j}$ and $l_{i,j}$ becomes $l'_{i,j} - r - z_{\{i,j\}} - z_{\{v_i,v_j\}} - x_{v_i,j} - x'_{v_i,j} - l_{i,j}$ (i.e., one more layer before the root r). It is easy to see that the length bound and size bound are still satisfied.

Assume there is a solution for the SLSN instance $(G', L, H_{k,k})$ with total cost less than $\left(\frac{5}{4} - \frac{\varepsilon}{4}\right) g(H_{k,k})$. The proof of existing a subgraph of G with $\varepsilon \cdot \binom{k}{2}$ edges is the same as Case 1, except the path between $l'_{i,j}$ and $l_{i,j}$ has one more layer.

4.5.2.3 Case 5: $\mathcal{H}_{2,k}$

For any $\varepsilon > 0$, assume there is a demand graph $H \in \mathcal{H}_{2,k}$ and a MULTI-COLORED DkS instance $G = (V, E)$ with coloring function c, factor ε, and parameter k. We create a unit-length and polynomial-cost SLSN instance G' as following.

We again use the length-weighted graph $G_{2,k}$ constructed in Section 4.3.2.3. We change the cost of edges in E_{22} to $4k^4(k-1)$, the cost of edges in $E_{12} \cup E_{xl}$ to $8k^4$, while keeping the cost equal to the length for the rest of the edges.

G' is again a graph that each edge $e \in E_{2,k}$ is replaced by a $length(e)$-hop path. Where the cost of edges is divided equally for each hop. The demands are the same as the demands in Section 4.3.2.3, and L is still 7. The construction still takes $|V||H|$. The function g is slightly different, where $g(H) = 16k^6 - 16k^5 - 10k^2 + 11k + 7|H| - 7 \cdot \mathbb{1}_{\{r_1,r_2\} \in H}$.

Using the same solution as in the proof of Lemma 4.3.10, we can see that, If there is a multi-colored clique of size k, then the SLSN instance has a solution with cost

$$\begin{aligned}
&7|H| - 7k^2 + 9k - 7 \cdot \mathbb{1}_{\{r_1,r_2\} \in H} \\
&+ k \cdot (4k^4(k-1) - 1) + k(k-1) \cdot (8k^4 - 1) + \binom{k}{2} \cdot (8k^4 - 4) \\
=&16k^6 - 16k^5 - 10k^2 + 11k + 7|H| - 7 \cdot \mathbb{1}_{\{r_1,r_2\} \in H},
\end{aligned}$$

because the cost of k edges in E_{22} in this solution is changed from 1 to $4k^4(k-1)$, the cost of $\binom{k}{2}$ edges in E_{12} in this solution is changed from 1 to $8k^4$, and

the cost of $k(k-1)$ edges in E_{xl} in this solution is changed from 4 to $8k^4$.

The other direction for the correctness is the following lemma.

Lemma 4.5.6. *Let S be an optimal solution for the* SLSN *instance (G', L) with demand graph $H \in \mathcal{H}_{2,k}$. If S has cost less than $\left(\frac{5}{4} - \frac{\varepsilon}{4}\right) g(H)$, then there is a subgraph of G with $\varepsilon \cdot \binom{k}{2}$ edges.*

Proof. For each $i, j \in [k]$ where $i \neq j$, let $P_{1,i,j}$ be a (arbitrarily chosen) path in S which connects r_1 and $l_{i,j}$, and $P_{2,i,j}$ be a (arbitrarily chosen) path in S which connects r_2 and $l_{i,j}$. Let $\mathcal{P}_1 = \{P_{1,i,j} \mid i, j \in [k], i \neq j\}$, and $\mathcal{P}_2 = \{P_{2,i,j} \mid i, j \in [k], i \neq j\}$. We can see that each of these paths must have exactly one edge between each two levels. Where paths in $\mathcal{P}_1$ have form $r_1 - z_{\{i,j\}} - z_{\{u,v\}} - x_{u,j} - l_{i,j}$ with $c(u) = i$, $c(v) = j$, and $\{u, v\} \in E$. The paths in $\mathcal{P}_2$ have form $r_2 - y_i - y_v - x_{v,j} - l_{i,j}$ with $c(v) = i$.

For each color $i \in [k]$, we can see that, in order to connect r_2 with all $l_{i,j}$, there must be at least one edge $\{y_i, y_v\} \in S \cap E_{22}$ with $v \in C_i$. Let $v_i = \arg\max_{v \in C_i} \left| \{y_i, y_v\} \cap \bigcup_{j \in [k] \setminus \{i\}} P_{2,i,j} \right|$ be the vertex which is in the most number of paths in $\mathcal{P}_2$. We will prove that the induced subgraph with vertex set $\{v_1, \ldots, v_k\}$ has at least $\varepsilon \cdot \binom{k}{2}$ edges.

For each $i \in [k]$, the edge $\{y_i, y_v\} \in S \cap E_{22}$ can only appear in at most $k-1$ different paths, which are $P_{2,i,j}$ where $j \in [k] \setminus \{i\}$. Because $\{y_i, y_{v_i}\}$ appears in most number of paths in $\mathcal{P}_2$, any $\{y_i, y_v\}$ other than $\{y_i, y_{v_i}\}$ can appear in at most $\frac{k-1}{2}$ different paths. Let $x = |S \cap E_{22}|$. Let $T = \{(i, j) \mid i, j \in [k], i \neq j, \{y_i, y_{v_i}\} \in P_{2,i,j}\}$. Then, $|T| \geq k(k-1) - (x-k) \cdot \frac{k-1}{2} = \frac{3}{2}k(k-1) - \frac{k-1}{2}x$.

We know that there is at least one different edge in E_{xl} for each $P_{1,i,j} \in \mathcal{P}_1$

where $(i,j) \in T$, thus $\left|E_{xl} \cap \bigcup_{(i,j)\in T} P_{1,i,j}\right| \geq |T|$. There is also at least one different edge in E_{xl} for each $P_{2,i,j} \in \mathcal{P}_2$, thus $\left|E_{xl} \cap \bigcup_{i,j\in[k],i\neq j} P_{2,i,j}\right| \geq k(k-1)$. Let $y = |S \cap E_{xl}|$, then

$$\begin{aligned}
&\left|E_{xl} \cap \bigcup_{(i,j)\in T} P_{1,i,j} \cap \bigcup_{i,j\in[k],i\neq j} P_{2,i,j}\right| \\
\geq &\left|E_{xl} \cap \bigcup_{(i,j)\in T} P_{1,i,j}\right| + \left|E_{xl} \cap \bigcup_{i,j\in[k],i\neq j} P_{2,i,j}\right| - |S \cap E_{xl}| \\
\geq &|T| + k(k-1) - y = \frac{5}{2}k(k-1) - \frac{k-1}{2}x - y.
\end{aligned}$$

For each $(i,j) \in T$, by looking at the form of $\mathcal{P}_1$ and $\mathcal{P}_2$, we know that $E_{xl} \cap P_{1,i,j}$ can not intersect with $P_{2,i',j'}$ with any $(i',j') \neq (i,j)$. And if $E_{xl} \cap P_{1,i,j}$ do intersect with $P_{2,i',j'}$, the intersection must be exactly one edge $\{x_{v_i,j}, l_{i,j}\}$. Therefore, let $T' = \{P_{1,i,j} \mid (i,j) \in T, \{x_{v_i,j}, l_{i,j}\} \in E_{xl} \cap P_{1,i,j} \cap P_{2,i,j}\}$, we have $|T'| \geq \frac{5}{2}k(k-1) - \frac{k-1}{2}x - y$.

Let $z = |S \cap E_{12}|$ and $R = \{\{v_i, v_j\} \mid P_{1,i,j} \in T', P_{1,j,i} \in T', P_{1,i,j} \cap E_{12} = P_{1,j,i} \cap E_{12}\}$. Because each path in T' has an edge in $S \cap E_{12}$, and any edge $(z_{c(u),c(v)}, z_{\{u,v\}}) \in S \cap E_{12}$ can appear in at most two paths $P_{1,c(u),c(v)}$ and $P_{1,c(v),c(u)}$ in T', we know that $|R| \geq |T'| - z$.

For any $\{v_i, v_j\} \in R$, because $\{x_{v_i,j}, l_{i,j}\} \in P_{1,i,j}$ and $\{x_{v_j,i}, l_{j,i}\} \in P_{1,j,i}$, by looking at the form of $P_{1,i,j}$ and the form of $P_{1,j,i}$, we know that $P_{1,i,j} \cap E_{12} = P_{1,j,i} \cap E_{12}$ can only be the edge $\{z_{\{i,j\}}, z_{\{v_i,v_j\}}\}$, which means $\{v_i, v_j\} \in E$. Therefore $R \subseteq E$. Thus the only thing left is to show that $|R| \geq \varepsilon \cdot \binom{k}{2}$.

Because $|H| \leq (k(k-1)+2)(k(k-1)+1)$, we have

$$\frac{k-1}{2}x + y + z \leq \frac{\left(\frac{5}{4} - \frac{\varepsilon}{4}\right) g(H)}{8k^4} \leq \left(\frac{5}{2} - \frac{\varepsilon}{2}\right) k(k-1).$$

Thus we have

$$\begin{aligned} |R| \geq & |T'| - z \geq \frac{5}{2}k(k-1) - \frac{k-1}{2}x - y - z \\ \geq & \frac{5}{2}k(k-1) - \left(\frac{5}{2} - \frac{\varepsilon}{2}\right) k(k-1) \geq \varepsilon \cdot \binom{k}{2}. \end{aligned}$$

Therefore the induced subgraph with vertex set $\{v_1, \ldots, v_k\}$ has at least $\varepsilon \cdot \binom{k}{2}$ edges. □

4.5.2.4 Case 6: $\mathcal{H}_k$

For any small constant $\varepsilon > 0$, we now want to construct a SLSN instance for a demand graph $H \in \mathcal{H}_k$ from a MULTI-COLORED DkS instance $(G = (V, E), c)$, factor ε, and parameter k. By definition of $\mathcal{H}_k$, for some $t \in [5]$ there is a graph $H^{(t)}$ of Case t which is an induced subgraph of H. We use Lemma 4.3.3 to find out the graph $H^{(t)}$. Let $(G^{(t)}, L)$ be the SLSN instance obtained from applying our reduction for case t from the MULTI-COLORED DkS instance $(G = (V, E), c)$. We want to construct a instance (G', c', L) with demand graph H, and makes sure that

- If the SLSN instance $(G^{(t)}, c^{(t)}, L)$ has a solution with cost $g^{(t)}(H^{(t)})$, then the SLSN instance (G', c', L) has a solution with cost $g(H)$.
- If the optimal cost of the SLSN instance (G', c', L) is less than $\left(\frac{5}{4} - \varepsilon\right) g(H)$, then the optimal cost of the SLSN instance $(G^{(t)}, c^{(t)}, L)$ is less than

$\left(\frac{5}{4} - \frac{\varepsilon}{4}\right) g^{(t)}(H^{(t)})$.

If there is such a construction, then

- If there is a multi-colored clique in G with size k, then the SLSN instance (G', c', L) has a solution with cost $g(H)$.
- If the optimal cost of the SLSN instance (G', c', L) is less than $\left(\frac{5}{4} - \varepsilon\right) g(H)$, then there is a induced subgraph of G with k vertices and at least $\varepsilon \cdot \binom{k}{2}$ edges.

Which is what we need for Theorem 4.5.4.

The graph G' is basically graph $G^{(t)}$ with some additional vertices and edges appeared in $H \setminus H^{(t)}$. We first increase the cost for all the edges in $G^{(t)}$ by multiplicative factor $\left\lceil \frac{L|H|}{\varepsilon} \right\rceil$. For each vertex v in H but not in $H^{(t)}$, we add a new vertex v to G'. For each edge $\{u, v\} \in H \setminus H^{(t)}$, we add a L-hop path between u and v to G', each new edge has cost 1. We set $g(H) = \left\lceil \frac{L|H|}{\varepsilon} \right\rceil \cdot g^{(t)}(H^{(t)}) + L \cdot (|H| - |H^{(t)}|)$. This is computable in $poly(|H|)$ time.

The construction still takes $poly(|V||H|)$ time, because the construction for the previous cases takes $poly(|V||H^{(t)}|)$ time and the construction for Case 6 takes $poly(|G^{(t)}||H|)$ time. Here $|H^{(t)}| \leq |H|$, and we know that $|G^{(t)}|$ is polynomial in $|V|$ and $|H^{(t)}|$.

If instance $(G^{(t)}, L, H^{(t)})$ has a solution with cost $g^{(t)}(H^{(t)})$, let the optimal solution be $S^{(t)}$. For each $e = \{u, v\} \in H \setminus H^{(t)}$, let the new L-hop path between u and v in G' be P_e. Then $S^{(t)} \cup \bigcup_{e \in H \setminus H^{(t)}} P_e$ is a solution to G' with cost $\left\lceil \frac{L|H|}{\varepsilon} \right\rceil \cdot g^{(t)}(H^{(t)}) + L \cdot (|H| - |H^{(t)}|) = g(H)$.

If instance instance (G', L, H) has a solution with cost less than $\left(\frac{5}{4} - \varepsilon\right) g(H)$, let the optimal solution be S. Since for each $e = \{u, v\} \in H \setminus H^{(t)}$, the only path between u and v in G' within the length bound is the new L-hop path P_e. Any valid solution must include all these P_e, which in total costs $L \cdot (|H| - |H^{(t)}|)$. In addition, for each demand $\{u, v\}$ which is also in $H^{(t)}$, any path between u and v in G' within the length bound will not include any new edge, because otherwise it will contain a L-hop path, and have length more than L. Therefore, $S \setminus \bigcup_{e \in H \setminus H^{(t)}} P_e$ is a solution to $G^{(t)}$ with cost less than

$$
\begin{aligned}
&\frac{1}{\left\lceil \frac{L|H|}{\varepsilon} \right\rceil} \left(\left(\frac{5}{4} - \varepsilon \right) g(H) - L \cdot (|H| - |H^{(t)}|) \right) \\
=&\frac{1}{\left\lceil \frac{L|H|}{\varepsilon} \right\rceil} \left(\left(\frac{5}{4} - \varepsilon \right) \left(\left\lceil \frac{L|H|}{\varepsilon} \right\rceil \cdot g^{(t)}(H^{(t)}) + L \cdot (|H| - |H^{(t)}|) \right) \right. \\
&\left. \quad - L \cdot (|H| - |H^{(t)}|) \right) \\
=&\left(\frac{5}{4} - \varepsilon \right) \cdot g^{(t)}(H^{(t)}) + \frac{L \cdot (|H| - |H^{(t)}|) \cdot \left(\frac{5}{4} - \varepsilon - 1 \right)}{\left\lceil \frac{L|H|}{\varepsilon} \right\rceil} \\
\leq&\left(\frac{5}{4} - \varepsilon \right) \cdot g^{(t)}(H^{(t)}) + \frac{L \cdot |H| \cdot \frac{1}{4}}{\frac{L|H|}{\varepsilon}} \\
\leq&\left(\frac{5}{4} - \varepsilon \right) \cdot g^{(t)}(H^{(t)}) + \frac{\varepsilon}{4} \\
\leq&\left(\frac{5}{4} - \frac{\varepsilon}{4} \right) g^{(t)}(H^{(t)}).
\end{aligned}
$$

Therefore Theorem 4.5.4 is proved.

4.5.3 Proof of Theorem 4.1.8:

If $\mathcal{C}$ is a recursively enumerable class, and $\mathcal{C} \nsubseteq \mathcal{C}_\lambda \cup \mathcal{C}^*$ for any constant λ, then for every $k \geq 2$, let H_k be the first graph in $\mathcal{C}$ where H_k is not a star and it has at least $2k^{10}$ edges. The time for finding H_k is $f(k)$ for some function f. From Lemma 4.3.3 we know that $H_k \in \mathcal{H}_k$, so that we can use Theorem 4.5.4 to construct the SLSN$_\mathcal{C}$ instance with demand H_k.

The parameter $p = |H_k|$ of the instance is only related with k, and the construction time is FPT from Theorem 4.5.4. Therefore this is a FPT reduction from the MULTI-COLORED DkS problem with parameter k and factor ε to the unit-length polynomial-cost SLSN$_\mathcal{C}$ problem with approximation factor $\left(\frac{5}{4} - \varepsilon\right)$. From Corollary 4.5.3, the unit-length polynomial-cost SLSN$_\mathcal{C}$ problem has no $\left(\frac{5}{4} - \varepsilon\right)$-approximation algorithm in $f(p) \cdot poly(n)$ time for any function f, assuming Gap-ETH. □

References

Hassin, Refael (1992). "Approximation schemes for the restricted shortest path problem". In: *Mathematics of Operations research* 17.1, pp. 36–42.

Lorenz, Dean H and Danny Raz (2001). "A simple efficient approximation scheme for the restricted shortest path problem". In: *Operations Research Letters* 28.5, pp. 213–219.

Chalermsook, Parinya, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan (2017). "From gap-ETH to FPT-inapproximability: Clique, dominating set, and more". In: *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*. IEEE, pp. 743–754.

Chitnis, Rajesh, Andreas Emil Feldmann, and Pasin Manurangsi (2017). "Parameterized Approximation Algorithms for Directed Steiner Network Problems". In: *arXiv preprint arXiv:1707.06499*.

Feldmann, Andreas Emil and Dániel Marx (2016). "The Complexity Landscape of Fixed-Parameter Directed Steiner Network Problems". In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*. Vol. 55. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 27:1–27:14.

Feldman, Jon and Matthias Ruhl (2006). "The directed Steiner network problem is tractable for a constant number of terminals". In: *SIAM Journal on Computing* 36.2, pp. 543–561.

Fellows, Michael R, Danny Hermelin, Frances Rosamond, and Stéphane Vialette (2009). "On the parameterized complexity of multiple-interval graph problems". In: *Theoretical Computer Science* 410.1, pp. 53–61.

Zeyu Zhang

zyzhang92@gmail.com
cs.jhu.edu/~zzy

Research Area	Theoretical computer science. Especially in approximation algorithms, graph algorithms, and algorithmic game theory
Education	**Johns Hopkins University**, Maryland, U.S. *08/2014-present* *Ph.D. candidate in Department of Computer Science* *Advised by prof.* Michael Dinitz *Overall GPA: 4.0/4.0* **Tsinghua University**, Beijing, China *08/2010-05/2014* *B.S.in Mathematical Sciences, Fundamental Science Class* *Affiliated student in Institute for Interdisciplinary Information Sciences (IIIS)* *Advised by prof.* Jian Li *Overall GPA: 85/100 (3.5/4.0)* **No. 2 High School of East China Normal University**, Shanghai, China *08/2007-05/2010*
Publications	Amy Babay, Michael Dinitz and **Zeyu Zhang**, "Characterizing Demand Graphs for (Fixed-Parameter) Shallow-Light Steiner Network", (FSTTCS 2018, https://arxiv.org/abs/1802.10566) Michael Dinitz, **Zeyu Zhang**, "Approximating Approximate Distance Oracles", (ITCS 2017, https://arxiv.org/abs/1612.05623) Michael Dinitz, **Zeyu Zhang**, "Approximating Low-Stretch Spanners", (SODA 2016, http://dx.doi.org/10.1137/1.9781611974331.ch59) Jian Li, **Zeyu Zhang**, "Ranking with Diverse Intents and Correlated Contents", (FSTTCS 2013, http://arxiv.org/pdf/1307.4518v2)
Teaching	TA: Introduction to Algorithms / Algorithms I (601.433/633, 600.463) *Fall 2015, Fall 2016, Fall 2018* - Given lectures on class about amortized analysis, union-find, recurrences and randomized quicksort TA: Approximation Algorithms (600.469/669) *Spring 2017* TA: Algorithmic Game Theory (601.436/636) *Spring 2018*

Zeyu Zhang

zyzhang92@gmail.com
cs.jhu.edu/~zzy

Programming Languages	Proficient: C/C++, Java, SQL Familiar with: Python, Pascal, Basic Tools and other Languages: Matlab, Mathematica, Latex, Lua
Selected Programming Projects	**Fault-tolerant distributed chat room** *Individual project in C* - Designed and constructed a reliable and efficient decentralized chat room service - Fully handled synchronization, packet loss, and crash recovery issues - Implemented functions such as post message, like/dislike, chat history, etc. **Time Series Topic Learning Model on Passive data for Parkinson's Disease** *Responsible for theory part and program logic, in Python* - Designed an algorithm to predict Parkinson's disease and its symptoms using daily gyroscope data collected by smartphone - Used an extended version of latest machine learning approach and figure out all the calculation
Work Experience	**Facebook, Inc.**, Menlo Park, U.S. *06/2018-09/2018* *Research scientist intern in Probability team* - Did predictions using different machine learning techniques - Reorganized data from many different databases using different tools - Used clusters to do preprocessing, training, and visualization all in parallel - Developed an internal predict server that also automatically update training results **Elite Marine Equipment & Engineering Inc.**, Nantong, China *08/2013-05/2014* *Product development consultant* - Mathematical modeling, flow simulation, and product designing **Citibank**, Shanghai, China *01/2012-02/2012* *Full-time intern in RMB settlement department* - Data analysis, as well as testing and debugging settlement systems
Previous Awards	First prize in National Olympiad in Informatics in Provinces, 2009 First prize in National Chemical Olympiad in Senior (Shanghai Province), 2009 Second prize in National Mathematical Olympiad in Senior (Shanghai Province), 2009
References	**Michael Dinitz** mdinitz@cs.jhu.edu **Xin Li** lixints@cs.jhu.edu